KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HARİTA MÜHENDİSLİĞİ ANABİLİM DALI

BİR NEHİR SİSTEMİ İÇİN EN UYGUN SU KALİTESİ İZLEME AĞININ TASARIMINI GENETİK ALGORİTMA VE YAPAY SİNİR AĞI KULLANILARAK GERÇEKLEŞTİRECEK, WEB TABANLI BİR YAZILIMIN GELİŞTİRİLMESİ

DOKTORA TEZİ

AmirAli VALIPOURYEKANI

KASIM 2021
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HARİTA MÜHENDİSLİĞİ ANABİLİM DALI

BİR NEHİR SİSTEMİ İÇİN EN UYGUN SU KALİTESİ İZLEME AĞININ TASARIMINI GENETİK ALGORİTMA VE YAPAY SİNİR AĞI KULLANILARAK GERÇEKLEŞTİRECEK, WEB TABANLI BİR YAZILIMIN GELİŞTİRİLMESİ

AmirAli VALIPOURYEKANI

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"DOKTOR (HARİTA MÜHENDİSİ)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 21 / 10 /2021
Tezin Savunma Tarihi : 22 / 11 /2021

Tez Danışmanı : Prof. Dr. Çetin CÖMERT

Trabzon 2021

ÖNSÖZ

Tez çalışmamın her aşamasında sabır, hoşgörü, manevi desteğini bir an olsun esirgemeyen, en zor zamanlarımda yoluma hep ışık olan ve yardım elini uzatan, kendime örnek edindiğim ve hayatım boyunca saygıyla hatırlayacağım kıymetli hocam Sayın Prof. Dr. Çetin CÖMERT'e sonsuz teşekkürlerimi sunarım. Komite üyelerim olarak çalışmamı başından itibaren takip eden ve doktora çalışmamın bu seviyelere gelmesinde emekleri olan Sayın Prof.Dr. Ayşe DALOĞLU ve DR. ÖĞR. ÜYESİ Esra TUNÇ GÖRMÜŞ'a ayrıca teşekkür ederim. Uzakta olmalarına rağmen tüm hayatım boyunca beni destekleyen, sevgileriyle yanımda olan ve beni motive eden değerli aileme sonsuz şükranlarımı sunarım.

AmirAli VALIPOURYEKANI

Trabzon 2021

TEZ ETİK BEYANNAMESİ

Doktora Tezi olarak sunduğum “Bir nehir sistemi için en uygun su kalitesi izleme ağının tasarımını GA ve Yapay Sinir Ağı kullanılarak gerçekleştirecek, web tabanlı bir yazılımın geliştirilmesi” başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Çetin CÖMERT’ ın sorumluluğunda tamamladığımı, verileri kendim topladığımı, analizleri ilgili laboratuvarlarda yaptığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 22.11.2021

AmirAli VALIPOURYEKANI

İÇİNDEKİLER

Sayfa No

Doktora Tezi

ÖZET

BİR NEHİR SİSTEMİ İÇİN EN UYGUN SU KALİTESİ İZLEME AĞININ TASARIMINI GENETİK ALGORİTMA VE YAPAY SİNİR AĞI KULLANILARAK GERÇEKLEŞTİRECEK, WEB TABANLI BİR YAZILIMIN GELİŞTİRİLMESİ

AmirAli VALIPOURYEKANI

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Harita Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Çetin CÖMERT
2021, 161 Sayfa, 18 Sayfa Ek

Su kalitesi alanında onlarca yıllık faaliyetlere rağmen, su kalitesi istasyonlarının yerleri halen birçok durumda ampirik yöntemlerle tespit edilmektedir. Bu çalışmanın temel amacı,NSGA-II algoritması, coğrafi bilgi sistemi teknolojisi ve açık kaynaklı araçları kullanarak nehirlerdeki su kalitesi izleme istasyonları ağının etkin tasarımı için web tabanlı bir uygulama geliştirmektir. Sonuç olarak, kullanıcılar bu programı masaüstü tabanlı uygulamaları kullanma sorunu ve bunları kullanmanın karmaşıklığı olmadan yalnızca internet ve web tarayıcısına erişimle programı kolayca kullanabilirler. Bu amaçla öncelikle nehir suyu kalite izleme istasyonları ağının tasarımında izlemenin genel hedefleri belirlendi ve fitness fonksiyonu tasarlandı. Ardından Visual Studio ortamında C# programlama dili kullanılarak bir web tabanlı uygulama gerçekleştirildi. Bu web tabanlı uygulama gerekli verileri Shapefile formatında alır ve NSGA-II uyguladıktan sonra web ve harita üzerinde su izleme istasyonlarının optimal yerlerini gösterir. Su kalitesi izleme istasyonlarının optimal yerlerini bulmak için,NSGA-II algoritma farklı nesil sayısında çalıştırıldı. 2500 neslinde Pareto optimal cevaplar elde edildi.NSGA-II sonuçları ile Analitik Hiyerarşik süreç ve Bulanık Mantık yöntemlerinin sonuçları karşılaştırıldığında, bu yöntemlerin sonuçlarının büyük ölçüde uyumlu olduğu gözlemlendi. Bir sonraki adımda, nehir suyu kalitesi izleme istasyonları optimal yerlerin bulunmasını hızlandırmak için GA ile optimize edilmiş çok katmanlı Perceptron sinir ağı (MLP) kullanıldı, Ve MLP sonuçları, NSGA-II sonuçları ile istatistiksel göstergelerle değerlendirildi. Momentum öğrenme kuralı ve Conjugate Gradiant öğrenme kuralı ile MLP sonuçları, 0.185 ve 0.163 hatalarla NSGA-II sonuçlarına benzer sonuçlar gösterdi.

Anahtar Kelimeler: Su kalitesi izleme ağı,NSGA-II ,MLP ,CBS, web tabanlı uygulama

PhD. Thesis

SUMMARY

DEVELOPMENT OF A WEB-BASED SOFTWARE FOR REALIZING THE OPTIMAL WATER QUALITY MONITORING NETWORK FOR A RIVER SYSTEM USING GENETIC ALGORITHM AND ARTIFICIAL NEURAL NETWORK

AmirAli VALIPOURYEKANI

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Geomatıc Engineering Graduate Program
Supervisor: Prof. Dr. Çetin CÖMERT
2021, 161 Pages, 18 Pages Appendix

Despite decades of activities in the field of water quality, the locations of water quality stations are still in many cases empirically determined. The main purpose of this study is to develop a web-based application for the effective design of a network of water quality monitoring stations in rivers using the NSGA-II, geographic information system, and open-source tools. As a result, users can easily use this program with only Internet and web browser access without the complexity of using desktop-based applications. For this purpose, in the design of the river water quality monitoring stations network, the general objectives of the monitoring were determined and the fitness function was designed. Then, a web-based application was implemented using the C# programming language in the Visual Studio environment. This web-based application receives the necessary data in Shapefile format and displays the optimal locations of water monitoring stations on the web and on the map after applying NSGA-II. To find the optimal locations of the water quality monitoring stations, the NSGA-II was run for different generations. Pareto optimal solutions were obtained in the 2500 generation. With the results of NSGA-II being compared with the results of Analytical Hierarchical Process and Fuzzy Logic methods, it was observed that the results of these methods were largely compatible. In the next step, a GA optimized multilayer Perceptron neural network (MLP) was used to accelerate the finding of optimal locations for river water quality monitoring stations, and the MLP results were evaluated by statistical indicators with the NSGA-II results. MLP results with momentum learning rule and Conjugate Gradient learning rule showed similar results to NSGA-II results with errors of 0.185 and 0.163.

Key Words: Water quality monitoring network, NSGA-II, MLP , GIS, web-based application

SEMBOLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DIZINI

CBS	: Coğrafi Bilgi Sistemleri
SOA	: Servis odaklı mimari
NSGA-II algorithm	: Non-dominated sorting genetic algorithm
SWMM	: Storm Water Management Model
SPARROW	: SPAtially Referenced Regression On Watershed attributes
TMDL	: Total maximum daily load
MSE	: Mean Square Error
RMSE	: Root Mean Square Error
MAE	: Mean absolute error
r	: Linear Correlation Coefficient
MLP	:Multi-layer perceptron (Çok Katmanlı Perceptron sinir ağı)
GA	: Genetik Algoritma
MOEA	:Multiobjective evolutionary algorithm
VEGA	:Vector Evaluated Genetic Algorithm
MOGA	: Multi objective Genetic Algorithm
NSGA	:Non-Dominated Sorting Genetic Algorithm
YSA	:Yapay sinir ağları

1. GİRİŞ

1.1. Giriş

Kuraklık krizi Dünyayı tehdit ediyor. Temiz su sıkıntısı, su kalitesinin izlenmesini eskisinden daha önemli hale getirmiştir (Dixon vd., 1999). Nüfusun hızla artması ve sanayileşme, suyun giderek azalması ve kirletilmesi sonucunu doğurmuştur. Son zamanlarda, kentleşme ve sanayileşmede artış ve su kirliliği geniş alanlar için bir tehdit haline gelmiştir (Dixon vd., 1999), Hem kamu ve hem politika yapıcılar nehirde izleme ağının tasarımı ve işletimi için iyileştirmeler talep edilmiştir. Dünya çapında havzaların tahribatı artmaktadır. Bunun sonucunda, iyi tasarlanmış bir izleme ağı, su kalitesi problemlerini ve uzun vadeli eğilimleri belirlemede çok önemli bir ön koşul haline gelmiştir. Su kalitesi izleme ağları ", su kalitesinin korunması, halihazırda uygulanan politikaların etkilerini değerlendirmek, ülke çapında nehirler deki mevcut su kalitesinin uzun ve kısa vadeli varyasyonlarını anlamak için temel bilgileri elde etmek ve yeni politikalar oluşturmak için işletilmektedir. " İzleme ağlarının tasarımına yönelik geçmiş yaklaşımlar çoğunlukla yapılandırılmış veya mantıksal bir tasarım metodolojisi olmaksızın keyfi bir temel üzerinde olmuştur. Yetersiz tasarlanmış su kalitesi izleme ağları, yalnızca hatalı değerlendirmelere değil aynı zamanda kötü yönetim ve kötü planlama değeri ile çok pahalı izleme verilerine de yol açabilir (Dixon vd., 1999).

1.2. Problemin Tanımı

Son zamanlarda, kentleşme ve sanayileşme artmış ve su kirliliği geniş alanlar için bir tehdit haline gelmiştir. Bunun sonucunda iyi ve verimli tasarlanmış bir izleme ağı, su kalitesi problemlerini ve uzun vadeli eğilimleri belirlemede çok önemli bir ön koşul haline gelmiştir. (Dixon vd., 1999). Bir nehirdeki su kalitesi izleme istasyonları ağının iyi ve

verimli bir şekilde tasarlanmasının amacı nehir suyu kalitesinin izlenmesi hedeflerine ve mevcut kısıtlama türlerine (operasyonel, sosyal, idari, yasal vb. kısıtlamalar) göre nehir suyunun kalitesi ve miktarı hakkında en fazla bilgiyi elde etmektir. Aynı zamanda, uygun maliyetli tasarımı sağlamak için ekonomik hususlar nehir suyu kalitesi izleme istasyonları ağının verimli tasarımına dahil edilmelidir. Bu nedenle, nehir suyu kalitesi izleme istasyonları ağının verimli tasarımı sorunu bir optimizasyon sorunudur. Çünkü bu tasarımda nehir suyunun kalitesi ve miktarı hakkında en fazla bilgiyi elde etmemiz ve aynı zamanda bu tasarımın maliyetlerini en aza indirmemiz gerekir. Optimizasyon için güçlü özellikleri nedeniyle GA hala ilk tercihtir. Bu yöntem kullanılarak birçok çalışma yapılmıştır. GA'ların optimal bir bölge bulma yeteneğini gösterdiler. Ancak bu çalışmalar GA'ları doğrudan CBS ile birleştirmedi. CBS mekansal analiz için güçlü bir araçtır. CBS yeteneklerini bazı istatistiksel analizler ve GA'lar gibi yumuşak hesaplamalar ile birleştirmek kullanıcıların karmaşık hesaplamalar yaparken mekansal kararlar için CBS özelliklerini kullanmasını sağlar. Başlangıçta, CBS masaüstü tabanlı uygulamalar kullanıyordu. Ancak kullanıcıların önce programı yüklemesi gerektiğinden kullanımı zordu. Günümüzde internetin hızla büyümesi ve bilgisayar, tablet ve akıllı telefon gibi çeşitli cihazlardan Web'e erişim ile; birçok uygulama geliştiricisi, web üzerinden erişilebilen hizmetleri kullanır. Web hizmetleri web uygulamalarının entegrasyonunu ve birlikte çalışabilirliğini ve makineler arasındaki etkileşimini sağlar. Servis odaklı mimari[1] (SOA) artık CBS uygulamalarının geliştirilmesi için ana modeldir. Geliştiriciler ve araştırmacılar, web hizmeti teknolojisini kullanarak web üzerinde mekansal uygulamalar geliştirdiler. Bu teknolojiyi kullanarak uygulamanın tüm detayları kullanıcıdan gizlenir ve kullanıcının yalnızca bir web tarayıcısına ihtiyacı vardır (Delipetrev, vd., 2014). CBS web çok ilginç çünkü internet ile belirli bir yeri yayınlayarak onu herkes için kolayca erişilebilir hale getiriyor ve şu anda herhangi bir CBS geliştiricisi için ilk tercihtir.Birçok araştırmacı bu tür CBS üzerinden çalışmıştır (Cömert ve Akinci, 2003), (Cömert, 2004), (Cömert ve Akinci, 2007), (Cömert vd., 2008), (Junghoon, 2011), (Fotheringham vd., 2005), (Urbieta, 2014), (Fustes, 2014), (Fotheringham vd., 2005), (Kulkarni vd., 2014). Bu çalışmanın temel amacı, GA , CBS teknolojisi ve açık kaynaklı araçları kullanarak nehirlerdeki su kalitesi izleme istasyonları ağının etkin tasarımı için web tabanlı bir uygulama geliştirmektir. Sonuç olarak, kullanıcılar bu programa masaüstü tabanlı uygulamalardaki

[1] Service-oriented architecture

kullanma sorunu ve bunları kullanmanın karmaşıklığı olmadan yalnızca internet ve web tarayıcılarının erişimiyle programı kolayca kullanabilirler.

1.3. Çalışmanın Amacı ve Hedefleri

Bu çalışmanın amacı GA, CBS teknolojisi ve açık kaynaklı araçları kullanarak nehirlerdeki su kalitesi izleme sensörleri ağının etkin tasarımı için web tabanlı bir uygulama geliştirmektir.

Projenin en temel hedefi, geliştirilecek yazılımın ülkemizde ve başka ülkelerdeki nehir ağları için kullanılabilir olmasını sağlamaktır. Herhangi bir akarsuya ait ve ilgili diğer veriler verildiğinde, en uygun sensör konumları anlık olarak belirlenebilecektir.

Diğer temel hedef nehir suyu kalitesi izleme istasyonlarının optimum yerlerini bulmak için GA ile optimize edilmiş YSA kullanımıdır.

1.4. Araştırma Soruları

Yukarıdaki hedeflere ulaşmak için bu araştırmanın ele aldığı araştırma problemleri aşağıda özetlenmiştir:

Su kalitesi ölçüm istasyonları ağının etkin tasarımında hangi hedefler ve kriterler dikkate alınmalıdır?

Bu hedef ve kriterlere göre hangi koşullar dikkate alınmalı, gerekli amaç fonksiyonları[1] veya Fitness fonksiyonu[2] nasıl tanımlanmalıdır?

Nehir suyu kalitesi izleme istasyonlarının optimum yerlerini bulmak için GA ile optimize edilmiş YSA kullanılabilir mi?

GA'ları CBS ile açık kaynak araçlarla doğrudan birleştirmek ve her iki yeteneğin aynı anda kullanılması için mümkün müdür ?

---

[1] Objective function
[2] Fitness function

2. BENZER ÇALIŞMALARIN İNCELENMESİ

Optimizasyon için güçlü özellikleri nedeniyle GA hala ilk tercihtir. Bu yöntem kullanılarak birçok çalışma yapılmıştır. GA'lar kullanılarak kablosuz sensör ağlarının uyarlanabilir tasarım optimizasyonu (Ferentinos vd., 2007), gelişmiş GAlar kullanılarak yapısal sağlık izleme için sensörlerin optimum yerleşimi (Guo, H. Y vd., 2004), GA'ları Kullanarak Sensör Yerleştirme Optimizasyonu (Spanache, S vd., 2004), GA'lar Kullanılarak Su Dağıtım Şebekelerinde Sızıntı Konumu için Optimum Sensör Yerleşimi (Casillas, M. V. vd., 2013), Orman yönetimi optimizasyonuna mekansal bir yaklaşım:CBS ve çok amaçlı GA'ları birbirine bağlamak (DUCHEYNE, Els I. vd., 2006), Otoyol Hizalamalarını Optimize Etmek için GA'ları ve Coğrafi Bilgi Sistemini Entegre Etme (Jha, M. K., & Schonfeld, P, 2000), GA'lar ve CBS Kullanılarak Turistlerle İlgili Kamu Tesislerinin Konumunun Değerlendirilmesi Yöntemi (Inoue, M., & Yamamoto, K. ,2013), GA'lar ve CBS ile Demiryolu Transit Güzergahlarının Optimize Edilmesi (Jha, M. K. vd., 2007), Kablosuz sensörler ağında çok amaçlı optimizasyon (LE BERRE vd., 2011), GA'ların optimal bir bölge bulma yeteneğini gösterdiler.

Su kalitesi izleme ağlarının optimal tasarımı ve verimliliklerinin iyileştirilmesi ile ilgili sorunlar 1970'lerden beri araştırma konusu olmuştur (Ning ve chang, 2002). Bu araştırmalarin devamında, su kalitesi izleme ağlarının optimal tasarımının izlenmesinin temel ilkeleri ve bağımsız izleme istasyonları için yer seçim kriterleri incelendi (Skalski ve Mackenzie, 1982; Lettenmaier vd., 1986; Smith ve McBride, 1990; Loftis vd., 1991; Esterby vd., 1992).

Sonraki çalışmalar, tam sayı programlama, çok amaçlı programlama, kriging teorisi ve optimizasyon analizi gibi çeşitli istatistiksel veya programlama yöntemlerini kullanarak su kalitesi izleme ağlarının etkili tasarımına odaklandı.

Hudak ve vd. (1995), bölgesel ölçekte yeraltı suyu kalitesini izlemek için ikili tam sayı matematiksel programlamayı kullandılar. Yeraltı suyu kirliliğini erken tespit etmek amacıyla, 1- Su kuyularının kirlilik kaynaklarından uzaklığı ve 2- Bu kuyudan kendi suyunu sağlayan nüfus miktarı kriterlerini kullanarak yeni bir yeraltı suyu izleme ağı

tasarladılar, ve ardından bir sıralama ölçeği kullanarak, yeni tasarlanan ağı önceden tasarlanmış ağ ile karşılaştırdılar. Yeni tasarlanan ağın önceden tasarlanan ağdan daha iyi performans gösterdiğini buldular.

Başka bir çalışmada, Dixon vd. (1999), su kalitesi izleme istasyonlarının yer seçimini optimize etmek için CBS[1], graf teorisi[2] ve benzetilmiş tavlama algoritmalarını[3] kullandılar. Bu çalışmanın amacı, nehirdeki kirliliğin kaynağını mümkün olan en kısa sürede ve en az bütçe ile bulup su kalitesi hakkinda en fazla bilgiyi elde etmekti. Bu çalışmada nehir sistemini göstermek için graf teorisi ve komşuluk matrisi[4] kullanılmıştır. Son olarak, benzetilmiş tavlama algoritması su kalitesi ölçüm istasyonlarının optimum yerlerini bulmak için kullanıldı.

Harmancıoğlu ve Alpaslan (1992), nehir suyu izleme ağının tasarımına yönelik temel yaklaşımları incelemiştir. Makalede, su kalitesi izleme ağlarının tasarımındaki mevcut durumu gözden geçirdiler. Ayrica su kalitesi izleme ağlarının tasarlanması sorununa yönelik mevcut yaklaşımların iyileştirilmesi için öneriler sundular. Makalede su kalitesi izleme ağ tasarımı konularının aslında optimizasyon sorunlarından biri olduğu belirtildi, Çünkü bu konularda görevimiz, belirtilen hedeflere en az bütçe ile su kalitesi hakkinda en fazla bilgiyi elde etmekti. Bu makale ayrıca nehir suyu izleme sisteminden elde edilen bilgilerin çeşitli amaçlarla kullanılabileceğini belirtmektedir. Bu nedenle, nehir suyu izleme ağının tasarımında bu farklı hedefler dikkate alınmalıdır.

Başka bir çalışmada Ning ve Chang (2002), bir su kalitesi izleme ağı tasarlamak için çok amaçlı karar temelli değerlendirme[5] kullandılar. Onlar bu çalışmada, birçok gerçek dünya meselesinin birden fazla hedef içerdiğini ve bu nedenle bu durumlarda çok amaçlı karar verme kullanılması gerektiğini belirttiler. Bu çalışmada, su kalitesi ölçüm istasyonlarının yeri ve sayısını seçmek için iki aşamalı bir analiz kullanılmıştır. İlk aşamada, su kalitesi izleme hedefleri bir anket kullanılarak belirlendi. Daha sonra her bir hedefin ağırlığı belirlendi. İkinci aşamada, çok amaçlı bir ağırlıklandırma programlama yöntemi kullanılarak nehir suyu kalitesi izleme istasyonlarının en uygun yerleri elde edildi.

---

[1] Coğrafi Bilgi Sistemi
[2] graph-theory
[3] Simulated annealing algorithm
[4] Adjacency matrix
[5] Multi-objective decision-based assessment

Başka bir çalışmada Yılmaz (2005), Türkiye'deki Gadiz Nehri üzerindeki su kalitesi izleme istasyonları ağını optimize etmek için bir GA kullandı. Bu çalışmada ilk olarak her istasyon bu hedeflere göre 1- Drenaj alanı 2- Nüfus 3- Sulama alanı 4- Gözlem sayısı 5- Gözlem süresinin uzunluğu 6- Su kalitesi değişkenleri sayısal değerlerle tanımlanmıştır. Daha sonra, değiştirilmiş çok amaçlı genel GA kullanılarak, 33 su kalitesi ölçüm istasyonundan 14 istasyon, su kalitesi ölçüm istasyonlarının optimum kombinasyonu olarak seçildiler. Bu çalışmada, su kalitesi izleme istasyonları ağının tasarımında GA kullanılmamış, sadece mevcut istasyonlardan en iyi su kalitesi izleme istasyonlarının seçilmesi için GA kullanılmıştır.

Başka bir çalışmada, Telci vd. (2009) nehir suyuna mümkün olan en kısa sürede kaza sonucu kirlenmeye neden olan kirleticileri tespit eden bir genetik algoritma kullanarak nehir suyunu izlemek için bir sistem tasarlamaya çalıştılar. Amerika'nin Georgia eyaletindeki Altamaha Nehri üzerinde örnek olay incelemelerini gerçekleştirdiler. İnsanları zararlı kirleticilere maruz kalmanın zararlı etkilerinden korumak için iki hedefe odaklandılar: 1. Kaza sonucu kirlenmenin erken tespiti 2. Nehir suyu kalitesi izleme sisteminin güvenilirliği. Bu araştırmada önerilen yöntem iki aşamaya dayanmaktadır: 1- Bir nehir ağındaki rastgele bir kirliliğin dinamik davranışının belirlenmesi ki, bu kısımda ikinci aşamada kullanılan veriler üretilir ve saklanır. 2. Bir optimizasyon yöntemine dayalı olarak kirlilik izleme konumlarının belirlenmesi. İlk adımda, tek bir rastgele kirliliğin dinamik davranışını belirlemek için SWMM[1] modeli kullanıldı. SWMM, öncelikle kentsel alanlarda tek olay veya uzun vadeli su akış miktarı ve kalitesi simülasyonları için kullanılır. SWMM, çalışma alanı giriş verilerini düzenlemek, hidrolojik, hidrolik ve su kalitesi simülasyonlarını çalıştırmak ve sonuçları çeşitli formatlarda görüntülemek için entegre bir ortam sağlar. Bunlar arasında renk kodlu drenaj alanı ve taşıma sistemi haritaları, zaman serisi grafikleri ve tabloları, profil grafikleri ve istatistiksel frekans analizleri bulunur. İkinci aşamada, GA iki temel amaç alınarak kullanılmıştır: 1- Kazara kirliliğin kaynağını tespit etmek için ortalama sürenin en aza indirilmesi 2- Nehir suyu izleme sisteminin güvenilirliğinin en üst düzeye çıkarılması. Sonuç olarak, su kalitesi izleme istasyonlarının en uygun yerleri bulundu.

Başka bir makalede Puri ve vd, (2017) nehir suyundaki patojenleri (bakteri yükü) modellemek ve tahmin etmek için çok amaçlı GA'yi kullandılar. Bakteriyel kontaminasyon

[1] Storm Water Management Model

genellikle aylık olarak veya bir fırtına sırasında veya sonrasında rastgele ölçülür. İzleme kısıtlamaları, ortalama yıllık bakteri kontaminasyonunu tahmin etmede belirsizlik yaratır. Bu da SPARROW[1] modeli sonuçlarında büyük hatalara yol açabilir. Bu çalışmada, uygun mekansal varyasyonlara sahip bir izleme istasyonları ağı seçmek için çok amaçlı bir GA kullanılmıştır. Bu araştırmada GA için seçilen hedefler 1- İzleme istasyonu sayısının en aza indirilmesi 2- En yüksek miktarda bakteri kontaminasyonunun olduğu yerler 3- Ölçülen verilerin belirsizliğinin en aza indirilmesi dir. En uygun istasyonların yerini bulduktan sonra, bu istasyonların verileri SPARROW modeli için kullanıldı. Bu çalışmanın sonuçları, GA'lar kullanılmadan yapılan önceki araştırmalarla karşılaştırıldı. Son olarak, GA yönteminden elde edilen verilerin kullanılmasının SPARROW modelinin sonuçlarını iyileştirdiği ve gerçek ölçüm verilerinden daha az hata gösterdiği gösterilmiştir. Bu çalışmada sadece nehirdeki patojenleri izlemek için uygun yerleri bulmak için GA kullanılmış, su kalitesini etkileyen diğer faktörler dikkate alınmamıştır.

Başka bir çalışmada, Parket vd. (2006), büyük bir nehir sisteminde etkili bir su kalitesi izleme ağı tasarlamak için tek amaçlı bir GA ve CBS kullanan entegre bir teknik önerdiler. Bu amaçla, önce su kalitesi izleme ağları için genel izleme hedefleri belirlenmiş ve buna dayalı olarak uygunluk fonksiyonu yukarıdaki hedeflere göre tanımlanır. Bu çalışmanın amacı, yakın zamanda yürürlüğe giren su yönetimi yönetmeliklerini ve teknolojilerini ve su kalitesi izlemenin geleneksel hedeflerini desteklemek için su kalitesi izleme ağlarının tasarlanması için bir çerçeve sağlamaktır. Bu amaçla 5 hedef belirlendi: 1- Su kalitesi standartlarına uyum 2- Kirlilik kaynaklarının izlenmesi 3- Su kullanımının izlenmesi 4- Su kalitesindeki değişikliklerin incelenmesi 6- Kirlilik yüklerinin tahmin edilmesi seçildi. Ayrıca bu çalışmada CBS'den GA giriş verilerinin hazırlanması ve analizi ayrıca GA'dan elde edilen verilerin görselleştirilmesi için kullandı. Tek amaçlı GA'yı uygulamak için Visual C ++ programlama dili ve GAlib GA kütüphanesi kullanıldı. Son olarak, optimum çözümler için Fitness seviyesi, uygun yakınsama belirlemek için GA'nın nesil sayısı kullanıldı; popülasyon boyutu, çaprazlama ve mutasyon gibi ana parametreler için duyarlılık analizi yapıldı.Böylece su kalitesi ölçüm istasyonları ağının en uygun konumu elde edildi. Ve bu yeni su kalitesi izleme ağı, önceki su kalitesi izleme ağı ile karşılaştırıldı. Sonuçlar, nehirde şu anda faaliyet gösteren 110 istasyondan sadece 35'inin yeni ağ tasarımındaki istasyonlarin yeri ile eşleştiğini gösterildi. Ve böylece, mevcut

[1] SPAtially Referenced Regression On Watershed attributes

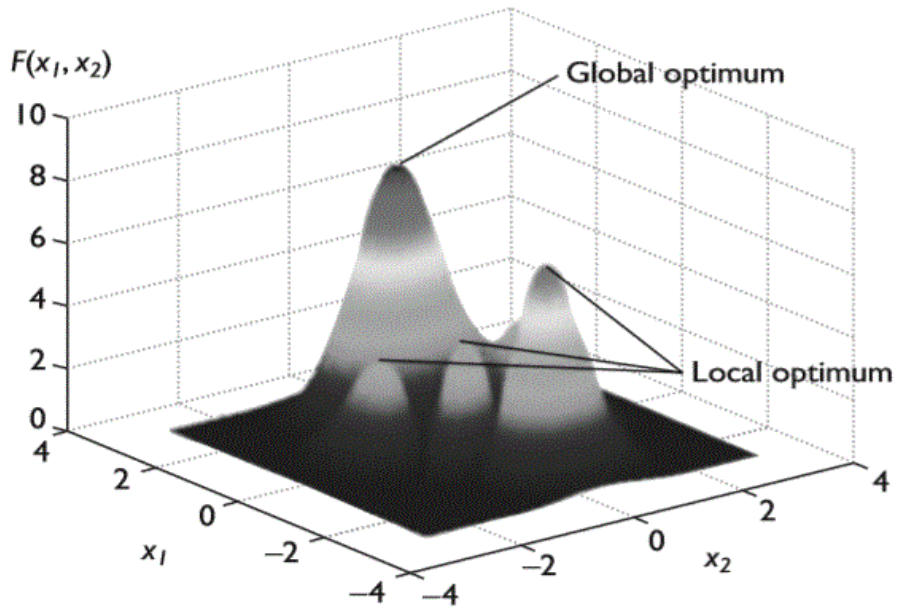
izleme ağının etkinliğinin dikkatlice değerlendirilmesi gerektiği gösterildi. Bu çalışmada önerilen yöntemin nehir suyu kalitesi izleme ağlarının optimum tasarımı için yararlı bir teknik olabileceği sonucuna varılmıştır. Bu çalışmada, su kalitesi izleme ağının tasarımı için makro düzeyde nehir suyu kalitesi izlemesi geleneksel ve modern hedefler temelinde yapılmış ve minör seviyesi sensör kurmak için uygun bir yer bulmaya hiç dikkat edilmemiştir. Ayrıca bu çalışmada su kalitesi ölçüm istasyonlarının kurulmasının maliyeti ve bütçesi ile ilgili konular dikkate alınmamıştır.

Başka bir çalışmada, GA'ların ve CBS'nin yeteneklerini aynı anda kullanmak için bir web tabanli uygulama tasarlandı. Bu çalışmada, GA'nın uygulanması, yerel koordinat sisteminin coğrafi koordinat sistemine dönüştürülmesi ve sonuçların coğrafi veritabanına gönderilmesi için MATLAB yazılımı kullanılmıştır. Ve sonunda sonuçlar web'de başarıyla görüntülendi (Handayanto vd., 2015).

3. GENEL BİLGİLER

3.1.Genetik Algoritma

GA'lar, bilgisayar bilimlerinde Doğal seçilim ve genetik ilkelerine dayanan arama ve optimizasyon algoritmaları dır. GA'lar ilk olarak 1970 yılında John Holland tarafından tanıtıldı. En uygun çözümü bulmak için yalnızca bir çözüm kullanan geleneksel yöntemlerin aksine, GA'lar en uygun çözümü bulmak için bir dizi çözüm kullanır. Bu nedenle, geleneksel yöntemlerden farklı olarak, GA'ların global optimum bulma ve lokal optimum takılıp kalmaktan kaçınma şansı daha yüksektir. lokal optima , yalnızca çözümün etrafındaki çözümlerden daha iyi bir çözüm olarak kabul edilen bir çözümdür, ancak global optimum, bir soruna yönelik tüm olası çözümlerden daha iyi çözüm olarak kabul edilir. Şekil 3.1 global ve lokal optimum gösterir. (Bölüm 3.1.1.4'te anlatılmıştır) (Sivanandam ve Deepa, 2008), (Li, Wei,2004), (Gen vd., 2008), (Eiben vd., 2003).



Şekil 3.1. Global ve lokal optimum (Abo-Hamad, W., & Arisha, A, 2011)

GA'lar, stokastik algoritmalar olarak kabul edilir ve kör arama[1] ve Hill climbing arama algoritmalarının bir kombinasyonudur. Hill climbing algoritması yerel arama ailesine ait matematiksel bir optimizasyon tekniğidir. Bu algoritma yinelemeli bir algoritmadır. Bu algoritma önce rastgele bir çözüm seçer ve daha sonra bu çözümdeki kademeli değişikliklerle en uygun çözümü bulmaya çalışır. Bu algoritma, en uygun çözümü aramak için bir çözümün etrafındaki uzayı kullandığından ve çözüm uzayının diğer kısımlarını aramadığından, birden fazla yerel optimuma sahip bir problemde yerel bir optimuma yakalanma olasılığı artar. GA, kör arama ve heuristic aramanın bir kombinasyonunu kullanır. Seçim, çaprazlama ve mutasyon gibi üç temel operatörü olan, GA da, kör arama mutasyon operatörü, heuristic arama ise seçim ve çaprazlama operatörleri tarafında gerçekleşir (Bölüm 3.1.1.3'te anlatılmıştır) (Sivanandam ve Deepa, 2008), (Li, Wei, 2004), (Gen vd., 2008), (Eiben vd., 2003).

GA yinelemeli[2] bir algoritmadır. Her yinelemeye bir nesil adı verilir ve soruna yönelik her olası çözüm bir kromozom veya birey olarak adlandırılır. Kromozomlar, gen adı verilen bileşenlerden oluşur ve her gen, sorunun nasıl kodlandığına bağlı olarak bitler veya sayılarla kodlanır. Bir dizi kromozom (çözümler veya bireyler) popülasyon olarak adlandırılır. Mutasyon ve çaprazlama, yeni kromozomlari oluşturmak için eski kromozomları kullanan GA operatörleri dir. Genetik algoritmalarda, kromozomların uyumu, fitness function veya object function kullanılarak belirlenir ve her bir kromozom puanlandırılır ve ardından seçim mekanizması ile en yüksek uyum puanına sahip kromozomlar sonraki nesile aktarılır. Bu seçilmiş kromozomlara ebeveyn (anne ve baba) adı verilir. Mutasyon ve çaprazlama operatörleri tarafından üretilen yeni kromozomlara yavru denir.

Optimizasyon problemlerini GA yöntemi ile çözerken ilk adımda problemin tüm çözümlerini kodlama zorundayız. Bu kodlanmış çözümlere kromozom denir. Bu kodlama, genetik algoritmanın başarısı ve uygulanmasında büyük etkisi vardır. Bu kodlamada, her bir kromozomun yalnızca bir çözümü temsil edecek şekilde olmalı ve soruna yönelik her potansiyel çözüm, bir kromozomla temsil edilmelidir. Kodlama aynı zamanda, kromozomların yapısındaki herhangi bir küçük veya büyük değişikliğin çözeltilerde eşit

[1] blind search
[2] iterative

şekilde görülebileceği şekilde olmalıdır ve bunun tersi de geçerlidir (Bölüm 3.1.1.6.2'te anlatılmıştır) (Sivanandam ve Deepa, 2008), (Li, Wei,2004), (Gen vd., 2008), (Eiben vd., 2003).

GA, tekrarlanan bir algoritmadır. Bu algoritma, rastgele bir kromozom seti seçerek başlar. Daha sonra, bu kromozomlar fitness function işlevi tarafından değerlendirilir ve her kromozoma bir uygunluk puanı verilir Bir sonraki adımda, GA'nın sonlandırma koşulu incelenir. Bu koşul karşılanırsa GA sona erer ve en yüksek uygun puanina sahip kromozomları (çözümler) nihai kromozomlar olarak gösterilir. GA'nın sonunun koşulu karşılanmazsa, bu GA yeni bir popülasyon, yani bir dizi kromozom, oluştururarak koşmaya devam edecektir. Yeni popülasyon üç aşamada oluşturulur: 1- Seçim: bir önceki nesilde en yüksek uyum puanını almış kromozomlar seçilir 2- çaprazlama: çaprazlama operatörü, ebeveynlerden (önceki adımda seçilen kromozomlar) ikisini birleştirmek ve bir yavru (yeni kromozom) üretmek için kullanılır 3- mutation: mutasyon operatörü yalnızca bir ebeveyni seçer ve genlerinde birkaç rastgele değişiklik yaparak yeni bir yavru üretir. Bir sonraki adımda, bu yavrular, ebeveynlerden yerini alır ve algoritma devam eder (Bölüm 3.1.1'te anlatılmıştır) (Sivanandam ve Deepa, 2008), (Li, Wei,2004), (Gen vd., 2008), (Eiben vd., 2003).

3.1.1.Basit Genetik Algoritma

GA, optimizasyon problemlerine yaklaşık çözümler bulan metaheuristic bir arama tekniğidir. GA'nın iki ayırt edici öğesi “birey” ve “popülasyon” dur. Birey, optimizasyon problemine tek bir aday çözüm iken popülasyon, optimizasyon sürecine dahil olan bir grup bireydir. Bireyin (fenotip) gözlenen özellikleri ham genetik bilgilerin (genotip) kodlanmış halidir. Genotip, GA'da kromozom olarak da adlandırılır (Sivanandam ve Deepa, 2008).

Çalıştırmadan önce GA'daki ilk adım, tüm olası çözümleri soyut bir temsil olan kromozomlara kodlamaktır. Bu adım aynı zamanda bir GA'nın en zor kısmı olabilir.

Temel GA pseudocode ve akış şemas şekıl 3.2 gösterdigi gibidir (Sivanandam ve Deepa, 2008):

1- ilk popülasyonu:

GA'yı bu aşamada başlatmak için ilk popülasyon genellikle rastgele oluşturulur. Her bir kromozom veya birey, soruna potansiyel bir cevaptır. Her bir kromozomun bileşenlerine gen denir ve bu genler, her sorunun özelliklerine göre bit veya sayılarla kodlanır. Bir dizi kromozom veya birey, popülasyon olarak adlandırılır.

2- Object fonksiyonu ile değerlendirme

Bu adımda, her kromozom fitness fonksiyonu tarafından değerlendirilir ve her kromozoma bir puan verilir. Fitness fonksiyonu veya object function, kromozomları değerlendiren bir işlevdir. Bu işlev, optimizasyonun amacına göre belirlenir.

3- Optimizasyon kriterleri karşılanıyor mu ?

Eğer optimizasyon kriterleri karşılanırsa GA durdurulur. Mevcut popülasyondaki en iyi kromozomlar nihai kromozomlar olarak seçilirler. Eğer optimizasyon kriterleri karşılanmazsa, GA yeni popülasyon üretmeye ile devam ediyor.

4- Yeni bir popülasyon üretmek:

Yeni bir popülasyon (yeni kromozomlar) üretmek için aşağıdaki adımlar gerçekleştirilir

- Seçim

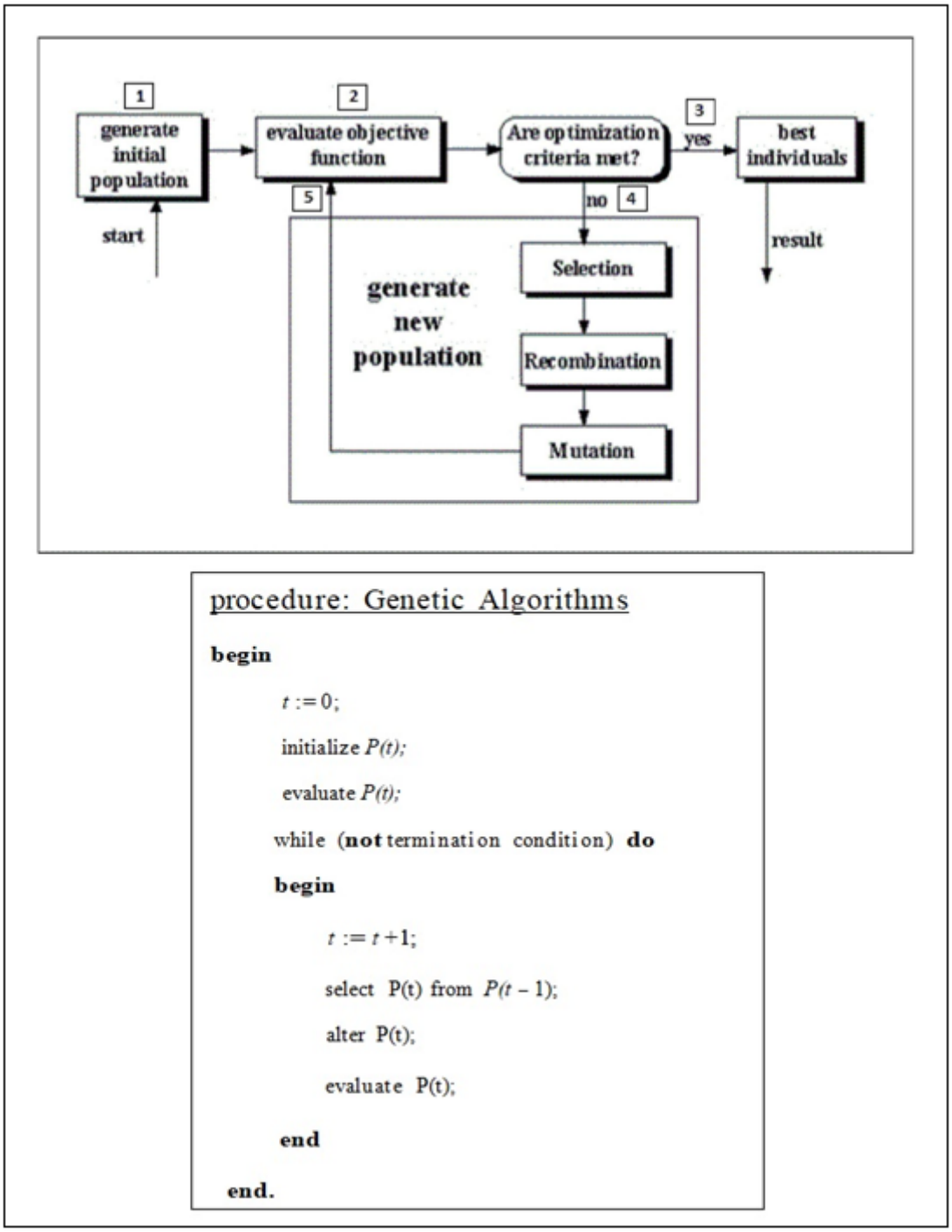
Seçim, fitness fonksiyonuna göre en yüksek puana sahip olan ve kromozomların GA'nın bir sonraki yinelemesine (nesline) aktarılmak üzere seçildiği bir mekanizmadır.

Burada, bir GA'nın her bir tekrarına bir nesil denir. Seçilen kromozomlara ebeveyn (anne ve baba) adı verilir. Ebeveynin mutasyon ve çaprazlama işlevleriyle üretilen yeni kromozomlarına yavru denir.

- Çaprazlama (Recombination veya Crossover): Ebeveynleri kullanarak yeni yavrular oluşturur.

- Mutasyon: Mutasyon yalnızca bir ebeveyn kromozomun yapısında (Genlerde) birkaç rastgele değişiklik yaparak yeni yavruları yaratır.

5- Sonunda bu yeni üretilen popülasyon, önceki popülasyonun yerini alır ve bu döngü devam eder.



Şekil 3. 2. GA pseudocode ve akış şeması (Sakawa, 2002), (Turabieh vd., 2019)

3.1.1.2. Genetik Algoritmanın Genel Yapısı

Genel olarak, bir GA'nın Michalewicz (1994) tarafından özetlendiği gibi beş temel bileşeni vardır:

1. Soruna yönelik olası çözümlerin genetik temsili.
2. Bir popülasyon yaratmanın bir yolu (ilk potansiyel çözümler seti).
3. Bir değerlendirme fonksiyonu, kromozomlari uygunluklarına göre derecelendirir ve her kromozoma uygunluk değeri atanir.
4. Yavruların genetik yapısını değiştiren genetik operatörler (çaprazlama, mutasyon, seçim).
5. GA'nın kullanıldığı parametre değerleri (popülasyon boyutu, genetik operatörleri uygulama olasılıkları vb).

3.1.1.3. Kullanma[1] ve Keşif[2]

Arama, çözüme götüren önceki adımların sırasını belirlemenin mümkün olmadığı bu tür problemler için evrensel problem çözme yöntemlerinden biridir. Arama, kör stratejiler veya heuristic stratejilerle yapılabilir (Bolc ve Cytowski, 1992). Kör arama stratejileri, sorunlu etki alanıyla ilgili bilgileri kullanmaz. Heuristic arama stratejileri en iyi arama yönleriyle birlikte arama hareketine rehberlik etmek için ek bilgileri kullanır. Arama stratejilerinde iki önemli konu vardır: En iyi çözümü kullanmak ve arama alanını keşfetmektir. (Booker, 1987). Michalewicz Hill climbing araştırması, rastgele arama ve genetik arama üzerine bir karşılaştırma yaptı. (Michalewicz, 1994). Hill climbing, arama alanının keşfini göz ardı ederek olası iyileştirme için en iyi çözümü kullanan bir strateji örneğidir. Rastgele arama, arama alanının ümit verici alanlarından yararlanma dan arama alanında arama yapan bir strateji örneğidir.

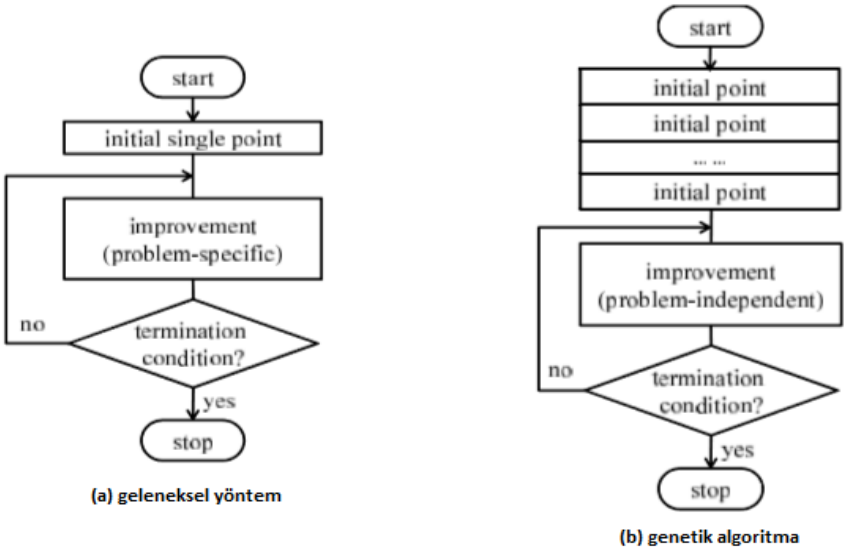
GA, arama alanının keşfi ve kullanma arasında dikkate değer bir denge sağlayabilen, yönlendirilmiş ve stokastik arama unsurlarını birleştiren genel amaçlı bir arama yöntemleri sınıfıdır. Genetik aramanın başlangıcında, geniş ölçüde rastgele ve çeşitli bir popülasyon vardır. Çaprazlama işlevi, tüm çözüm uzayını keşfetmek için geniş çaplı arama yapma eğilimindedir. Uygun çözümleri geliştikçe, çaprazlama operatörü, her birinin yakınında

[1] Exploitation
[2] Exploration

keşif sağlar. Başka bir deyişle bir çaprazlama geçişin ne tür aramalar (kullanma veya keşif) gerçekleştireceği, operatörün kendisi tarafından değil genetik sistemin çevresi (nüfusun çeşitliliği) tarafından belirlenecektir. Ek olarak, basit genetik operatörler genel amaçlı arama yöntemleri olarak tasarlanırlar, esasen kör bir arama yaparlar ve gelişmiş bir yavru vermeyi garanti edemezler.

3.1.1.4. Nüfusa Dayalı Arama

Genel olarak, optimizasyon problemlerini çözmek için bir algoritma, asimptotik olarak optimal çözüme yakınsayan bir dizi hesaplama adımıdır. Çoğu klasik optimizasyon yöntemi, objektif fonksiyonun gradyanına veya daha yüksek dereceli türevlerine dayanan deterministik bir hesaplama dizisi üretir. Yöntemler, arama alanında tek bir noktaya uygulanır. Nokta daha sonra Şekil 3.3'de gösterildiği gibi yinelemeler yoluyla en derin alçalan yönde kademeli olarak geliştirilir. Bu noktadan noktaya yaklaşım, yerel optimizasyonlarda başarısızlık riskini içerir. GA, potansiyel çözümlerin bir popülasyonunu koruyarak çok yönlü bir arama gerçekleştirir. Popülasyondan-popülasyona yaklaşımı, yerel optimumdan arama kaçışını yapmak için umut vericidir. Popülasyon simüle edilmiş bir evrim geçirir: Her nesilde nispeten iyi çözümler yeniden üretilirken, nispeten kötü çözümler ölür. GA, çoğaltılacak birini ve ölecek birini seçmek için olasılıksal geçiş kurallarını kullanır ve böylece aramalarını arama alanının bölgelerine doğru olası iyileştirmelerle yönlendirir (Gen vd., 2008).



Şekil 3.3. Geleneksel ve genetik yaklaşımların karşılaştırılması(Gen vd, 2008)

3.1.1.5. Başlıca Avantajları

GA, yeni bir optimizasyon tekniği olarak potansiyelleri açısından büyük ilgi görmüştür. Optimizasyon problemlerine GA uygularken üç büyük avantaj vardır (Gen vd., 2008):

1. Uyarlanabilirlik: GA, optimizasyon problemleriyle ilgili çok fazla matematiksel gereksinime sahip değildir. Evrimsel doğası gereği, GA problemin özel iç işleyişine bakmaksızın çözüm arayacaktır. GA, kesikli, sürekli veya karışık arama alanları üzerinde tanımlanan her türlü nesnel işlevi ve her türlü kısıtlamayı, yani doğrusal veya doğrusal olmayan , işleyebilir.

2. Sağlamlık[1]: Evrim operatörlerinin kullanılması, GA'yı genel bir arama gerçekleştirmede (olasılıkla) çok etkili hale getirirken, çoğu geleneksel buluşsal yöntem genellikle yerel bir arama gerçekleştirir. GA'nın en uygun çözümü bulmada ve hesaplama çabasını azaltmada diğer geleneksel buluşsal yöntemlere göre daha verimli ve daha sağlam olduğu birçok çalışma tarafından kanıtlanmıştır.

[1] Robustness

3. Esneklik[1]: GA, belirli bir problem için etkili bir uygulama yapmak üzere domain bağımlı sezgisel yöntemlerle hibritleşmek için büyük esneklik sağlar.

3.1.1.6. Genetik Algoritmaların Uygulanması

GA'nın uygulanmasında birkaç bileşen dikkate alınmalıdır. İlk olarak, çözümlerin genetik temsiline karar verilmelidir (yani kodlama); ikincisi, çözümleri değerlendirmek için bir uygunluk fonksiyonu verilmelidir. (yani, kod çözme) Üçüncü olarak, çaprazlama operatörü, mutasyon operatörü ve seçim yöntemleri gibi genetik operatörler tasarlanmalıdır. Son olarak, kısıtlı optimizasyona GA uygulamak için gerekli bir bileşende kısıtlamaların nasıl ele alınacağıdır. Çünkü kromozomları manipüle etmek için kullanılan genetik operatörler genellikle uygun olmayan yavrular verir.

3.1.1.6.1. Kodlama Konusu

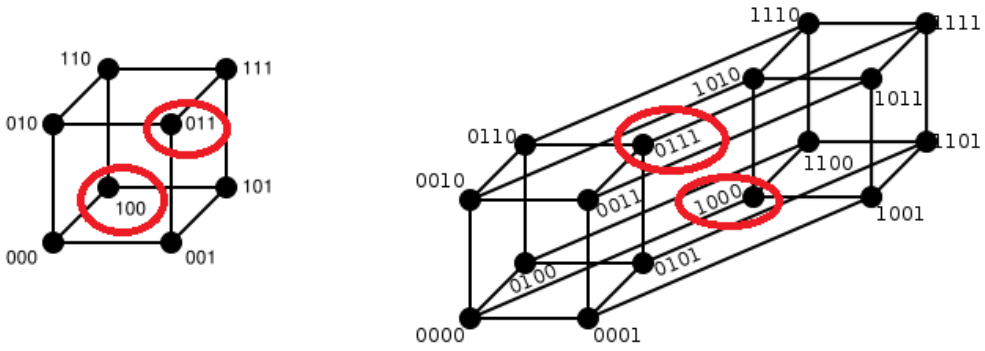
Belirli bir sorunun çözümünün bir kromozoma nasıl kodlanacağı, GA için önemli bir konudur. Bu konu birçok yönden araştırılmıştır ve her biri aşağıda açıklanmıştır.

3.1.1.6.1.1. Kodlamaların Sınıflandırılması

Holland'ın çalışmasında, kodlama ikili dizeler (Holland, 1992) kullanılarak gerçekleştirilir. Fonksiyon optimizasyon problemleri için ikili kodlamanın, büyük bir Hamming uzaklığı'na (Hamming uzaklığı aynı uzunluktaki iki dizenin farklı elemanlarının sayısıdır. Hamming uzaklığı adı Richard Hamming'in adından gelir) sahip bir çift kodlamanın fenotip uzayında minimum mesafeli noktalara ait olduğu olgusunu tanımlayan Hamming cliffs varlığından dolayı ciddi dezavantajlara sahip olduğu bilinmektedir. Örneğin, 0111 ve 1000 ya 011 ve 100 çifti, fenotip uzayındaki komşu noktalara (minimum Öklid mesafelerinin noktaları) aittir.Ancak genotip uzayında maksimum Hamming

[1] Flexibility

mesafesine sahiptir. Hamming cliff geçmek için tüm bitlerin bir kerede değiştirilmesi gerekir. Çaprazlama ve mutasyonun onu geçmek için ortaya çıkma olasılığı çok küçük olabilir. Bu anlamda, ikili kod fenotip uzayındaki noktaların yerini korumaz (Gen vd., 2008).



Şekil 3.4. Hamming uzaklığı.(Wikipedia)

Pek çok gerçek dünya uygulaması için, çözümlerini ikili kodlama ile temsil etmek neredeyse imkansızdır. GA'nın etkin bir şekilde uygulanabilmesi için belirli sorunlar için çeşitli kodlama yöntemleri oluşturulmuştur. Bir genin allelleri olarak ne tür semboller kullanıldığına göre kodlama yöntemleri şu şekilde sınıflandırılabilir (Gen vd., 2008):

- İkili kodlama
- Gerçek sayı kodlaması
- Tam sayı / değişmez permütasyon kodlaması
- genel bir veri yapısı kodlaması

Gerçek sayı kodlama fonksiyonu optimizasyon problemleri için en iyisidir. Fonksiyon optimizasyonları ve kısıtlı optimizasyonlar için gerçek sayı kodlamasının ikili veya Gray kodlama dan daha yüksek performansa sahip olduğu geniş çapta onaylanmıştır. Gerçek sayı kodlama yöntemi için genotip uzayının topolojik yapısı, fenotip uzayınınki ile aynı olduğundan, geleneksel yöntemlerden bazı yararlı teknikleri ödünç alarak bazı etkili genetik operatörler yaratmak bizim için kolaydır. Tam sayı veya değişmez permütasyon (literal permutation) kodlaması, kombinatoryal optimizasyon problemleri için uygundur.

Kombinasyonel optimizasyon problemlerinin özü, en iyi permütasyonu aramaktır veya bazı kısıtlamalara tabi bazı öğelerin kombinasyonu, literal permütasyon kodlaması, bu tür bir sorunla başa çıkmanın en makul yolu olabilir. Daha karmaşık gerçek dünya problemleri için, problemin doğasını yakalamak için bir genin aleli olarak uygun bir veri yapısı önerilmektedir. Bu gibi durumlarda, bir gen n-ary veya daha karmaşık bir veri yapısı olabilir. Kodlamaların yapısına göre, kodlama yöntemleri aşağıdaki iki tipte sınıflandırılabilir (Gen vd., 2008) :

• Tek boyutlu kodlama

• Çok boyutlu kodlama

Çoğu durumda, tek boyutlu şifreleme yöntemi kullanılır. Bununla birlikte, gerçek dünyadaki birçok sorunun çok boyutlu yapıların çözümleri vardır. Bu çözümleri temsil etmek için çok boyutlu bir kodlama yöntemi benimsemek doğaldır. Hangi tür içeriklerin kodlandığına göre kodlama yöntemleri de şu şekilde bölünebilir (Gen vd., 2008):

• Yalnızca çözüm

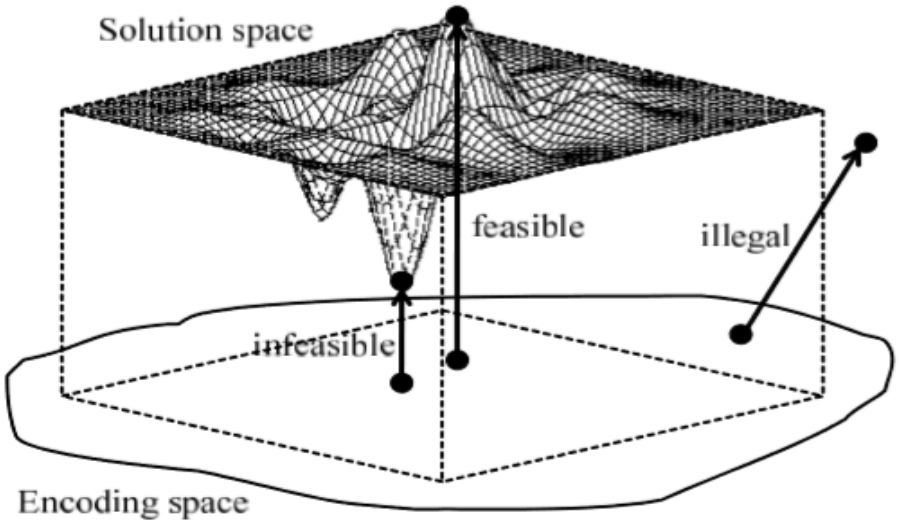
• Çözüm + parametreler

GA uygulamasında, ilk yöntem, belirli bir probleme uygun bir kodlama tasarlamak için yaygın olarak benimsenir. İkinci yol ise Rechenberg (Rechenberg,1973) ve Schwefel (Schwefel,1995) tarafından evrim stratejilerinde kullanılmaktadır. Bir birey iki bölümden oluşur: İlk bölüm belirli bir problemin çözümüdür ve strateji parametreleri olarak adlandırılan ikinci bölüm, mutasyon için normal dağılımın varyanslarını ve kovaryanslarını içerir. Strateji parametrelerini bireylerin temsiline dahil etmenin amacı, onlara evrimsel işleçler uygulayarak bu parametrelerin evrimsel olarak kendi kendine adaptasyonunu kolaylaştırmaktır. Daha sonra çözüm uzayında ve strateji parametrelerinde birlikte arama yapılacaktır. Bu şekilde, gelişigüzel koşullar altında mutasyon parametrelerinin uygun bir ayarlaması ve çeşitliliği sağlanmalıdır.

3.1.1.6.1.2. Olursuz[1] ve Yasadışılık[2]

[1] Infeasibility
[2] Illegality

GA, alternatif olarak iki tür alan üzerinde çalışır: Kodlama[1] Uzayı ve Çözüm Uzayı veya diğer sözcüklerde, genotip uzayı ve fenotip uzayı. Genetik operatörler, genotip uzayı üzerinde çalışırken, değerlendirme ve seçim fenotip uzayı üzerinde çalışır. Doğal seçilim, kromozomlar ile kodu çözülen çözümlerin[2] performansı arasındaki bağlantıdır. Genotip uzayından fenotip uzayına yapılan haritalama[3], GA'nın performansı üzerinde önemli bir etkiye sahiptir. Haritalama ile ilgili en belirgin sorun, bazı bireylerin belirli bir soruna uygulanabilir olmayan çözümlere karşılık gelmesidir. Bu sorun, kısıtlı optimizasyon problemleri ve kombinasyonel optimizasyon problemleri için çok ciddi hale gelebilir. Şekil 3.5'te gösterildiği gibi, iki temel kavram arasında ayrım yapmalıyız: "olurlu" / "olursuz" ve yasadışılık. Bunlar literatürde genellikle yanlış kullanılırlar. olursuz bir kromozomdan kodu çözülen bir çözümün belirli bir problemin "olurlu" bölgesinin dışında olduğu, yasadışılık ise bir kromozomun belirli bir probleme bir çözüm sunmadığı durumdur (Gen vd, 2008). Örneğin bizim çalışmamızda su kalitesi sensörlerinin çalışabilmesi için nehir suyuna yerleştirilmesi gerekmektedir. Sonuç olarak olurlu uzay nehir çevresi, olursuz uzay Dejirmandere havzasının nehir çevresinde olmayan diğer alanları, yasadışı uzay Dejirmandere havzası dışındaki alanlar denilmektedir.



Şekil 3.5. İmkansızlık ve yasadışılık. (Gen vd., 2008).

[1] Encoding
[2] Decoded solutions
[3] Mapping

Kromozomların imkansızlığı, kısıtlanmış optimizasyon probleminin doğasından kaynaklanmaktadır. Hangi yöntemler kullanılırsa kullanılsın, geleneksel yöntemler veya GA'lar, kısıtlamaları ele almalıdır. Çoğu optimizasyon problemleri için, olurlu bölge, bir eşitlikler veya eşitsizlikler sistemi olarak temsil edilebilir. Bu tür durumlarda, olursuz kromozomları ele almak için ceza yöntemleri kullanılabilir (Gen & Cheng,1997). Kısıtlı optimizasyon problemlerinde optimum, tipik olarak, olurlu ve olursuz uzayler arasındaki sınırda gerçekleşir.

Kromozomların yasadışılığı, kodlama tekniklerinin doğasından kaynaklanmaktadır. Pek çok kombinatoryal optimizasyon problemi için, probleme özgü kodlamalar kullanılır ve bu tür kodlamalar genellikle basit bir tek kesim noktalı çaprazlama işlemiyle yasadışı yavrular verir. Yasadışı bir kromozomun dekod etmeden, ceza teknikleri bu duruma uygulanamaz. Onarım teknikleri genellikle yasadışı bir kromozomu yasal olana dönüştürmek için benimsenir. Orvosh ve Davis (Orvosh, Davis, 1994) birçok kombinatoryal optimizasyon problemi için, olursuz veya yasadışı bir kromozomu tamir etmenin nispeten kolay olduğunu ve onarım stratejisinin, “reddetme” veya “cezalandırma” gibi diğer stratejileri gerçekten geride bıraktığını göstermiştir.

3.1.1.6.1.3. Kodlamaların Özellikleri

Yeni bir kodlama yöntemi verildiğinde, genellikle kodlama ile etkili bir inşa edip edemeyeceğimizi incelemek gerekir. Bir kodlamayı değerlendirmek için birkaç ilke önerilmiştir (Gen ve Cheng,1997), (Raidl ve Julstrom, 2003):

Özellik 1 (Boşluk): Kromozomlar aşırı miktarda hafıza gerektirmemelidir.

Özellik 2 (Zaman): Kromozomlar üzerinde değerlendirme, rekombinasyon ve mutasyon gerçekleştirmenin zaman karmaşıklığı daha yüksek bir mertebe olmamalıdır.

Özellik 3 (Fizibilite): Bir kromozom, uygulanabilir bir çözüme karşılık gelir.

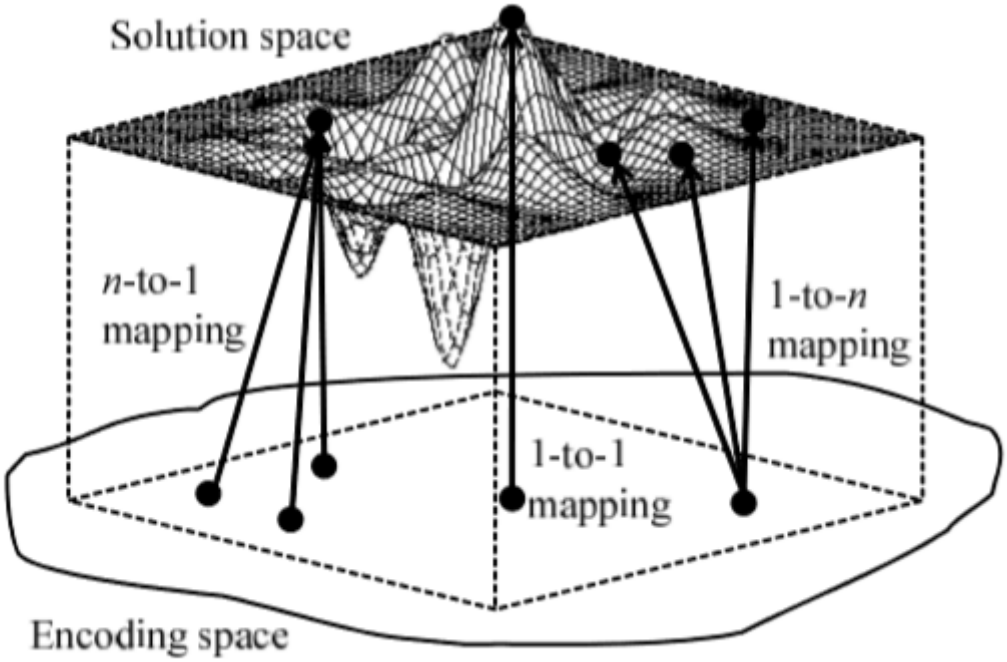
Özellik 4 (Yasallık): Bir kromozomun herhangi bir permütasyonu bir çözüme karşılık gelir.

Özellik 5 (Tamlık): Herhangi bir çözümün karşılık gelen bir kromozomu vardır.

Özellik 6 (Benzersizlik): Kromozomlardan çözümlere (kod çözme) eşleme, aşağıdaki üç durumdan birine ait olabilir (Şekil 3.6): 1'e 1 haritalama, n'den 1'e haritalama ve 1'den n'ye haritalama. 1'e 1 eşleme, üç durum arasında en iyisidir ve 1'den n'ye eşleme en istenmeyen olanıdır.

Özellik 7 (Kalıtılabilirlik): Basit çaprazlama yavruları (yani, tek kesme noktalı çaprazlama), ebeveynlerinin temel özelliklerini birleştiren çözümlere karşılık gelmelidir.

Özellik 8 (Lokalite): Kromozomdaki küçük bir değişiklik, karşılık gelen çözümde küçük bir değişiklik anlamına gelmelidir.



Şekil 3.6. Kromozomlardan çözümlere haritalama. (Gen vd., 2008).

3.1.1.6.2. Başlatma

Genel olarak, ilk popülasyonu oluşturmanın iki yolu vardır, yani, problemin sınırlarını ve / veya sistem kısıtlamalarını karşılarken heuristic başlatma ve rastgele başlatmadır. Heuristic başlatmanın ortalama uyum nispeten yüksek olmasına rağmen, GA'nın çözümleri daha hızlı bulmasına yardımcı olabilir, çoğu büyük ölçekli problemde, örneğin ağ tasarım problemlerinde, heuristic yaklaşım, çözüm uzayının sadece küçük bir bölümünü keşfedebilir ve nüfustaki çeşitlilik eksikliği nedeniyle küresel optimal çözümler bulmak zordur. Genellikle ilk popülasyonu oluşturmak için kromozoma bağlı olarak bir kodlama prosedürü tasarlamamız gerekir (Gen vd., 2008).

3.1.1.6.3. Uygunluk Fonksiyonu:

Uygunluk işlevi (optimizasyon problemlerinde amaç fonksiyonu da adlandırılır) GA'nın en önemli bileşenidir. Uygunluk işlevi, aday çözümlerin iyiliğini ölçer ve seçim için bir temel oluşturur ve böylece iyileştirmeleri basitleştirir. Başka bir deyişle, uygunluk işlevi, belirli bir sorun için iyileştirmenin ne anlama geldiğini tanımlar (Eiben vd., 2003).

3.1.1.6.4. Genetik Operatörler

GA ilerlediğinde, arama alanında keşif ve Kullanım bir denge sağlamak için hem en uygun çözüme giden arama yönü hem de arama hızı önemli faktörler olarak düşünülmelidir. Genel olarak, GA araştırmasından kaynaklanan birikmiş bilgilerin kullanımı, seçim mekanizması tarafından yapılırken, arama alanının yeni bölgelerine keşif, genetik operatörler tarafından yapilir (Eiben vd., 2003).

Genetik operatörler, her nesilde yeni yavrular yaratmak için genlerin kalıtım sürecini taklit ederler. Operatörler, temsil sırasında bireylerin genetik kompozisyonunu değiştirmek için kullanılır. Temelde, operatörler rastgele bir arama yaparlar ve gelişmiş bir yavru vermeyi garanti edemezler. Üç yaygın genetik operatör vardır: Çaprazlama, mutasyon ve seçim (Eiben vd., 2003) , (Sivanandam ve Deepa, 2008).

3.1.1.6.4.1. Çaprazlama

Çaprazlama, çiftleşme havuzundan iki ebeveyni alan süreçtir ve başka özlliklerde bir yavru elde etmek için seçim sürecinden sonra oluşturulur. Çaprazlama, aşağıdaki gibi üç adımdan oluşur (Sivanandam ve Deepa, 2008):

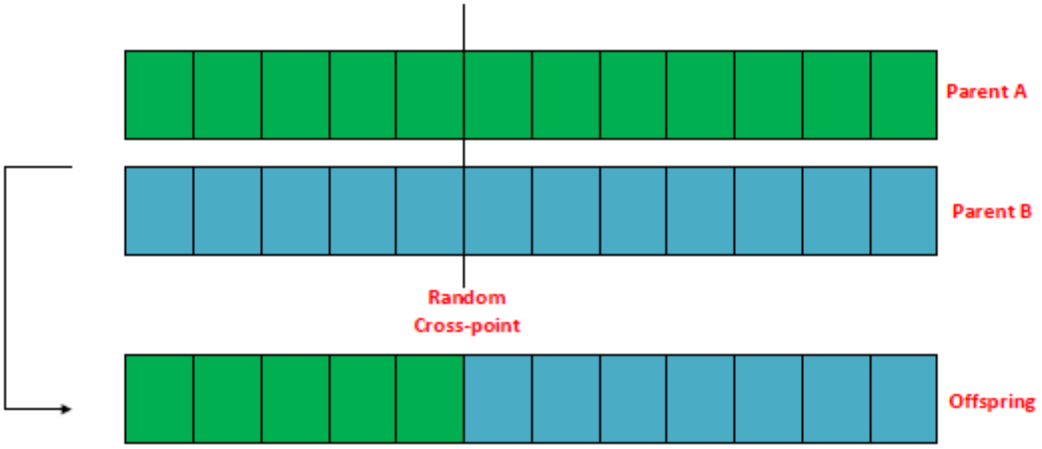
(Çiftleşme havuzu, mevcut popülasyonda en yüksek uygunluğa sahip aday çözümlerden oluşur. Çiftleşme havuzuna dahil edilen çözümlere ebeveyn adı verilir.)

1. Çiftleşme havuzundan rastgele iki birey ebeveyn olarak seçilir.
2. Aynı pozisyondaki her bir birey için rastgele bir çapraz nokta seçilir.
3. Son olarak, çapraz nokta senden sonra iki kişi arasında değerler değiş tokuş edilir.

Çeşitli Çaprazlama geçiş teknikleri mevcuttur ve bunlardan bazıları aşağıdaki şekilde tartışılmıştır (Eiben vd., 2003) , (Sivanandam ve Deepa, 2008):

-Tek Noktalı Çaprazlama:

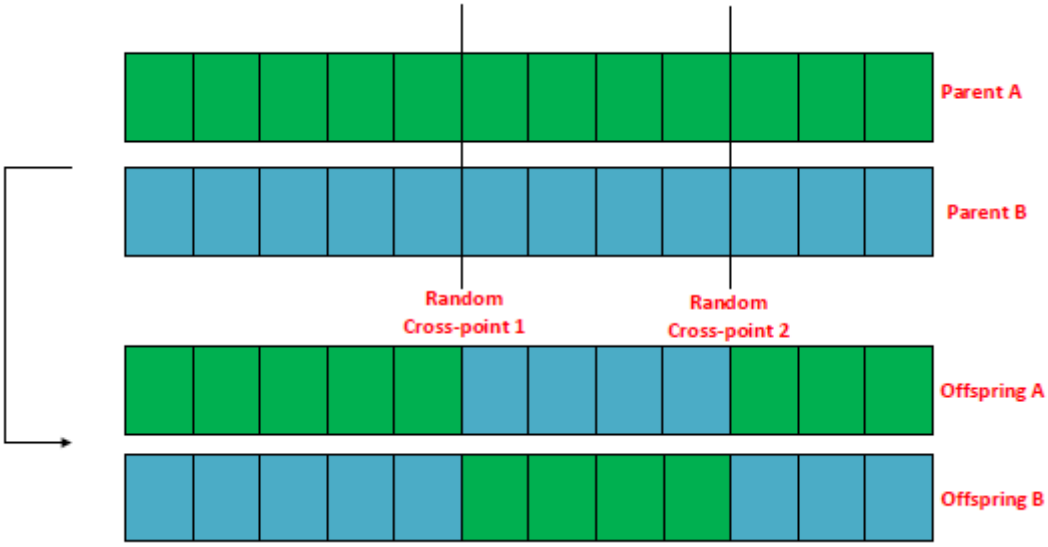
Bu yöntem, geleneksel GA'da kullanılan en yaygın operatörlerden biridir.İki ebeveynin kesildiği rastgele bir çapraz nokta kullanır.Ardından, çapraz noktanın yanındaki genomlar, yeni bir çocuk üretmek için değiştirilir.



Şekil 3.7. Tek Noktalı Çaprazlama (S.S.Noorian, 2015).

-İki Noktalı Çaprazlama:

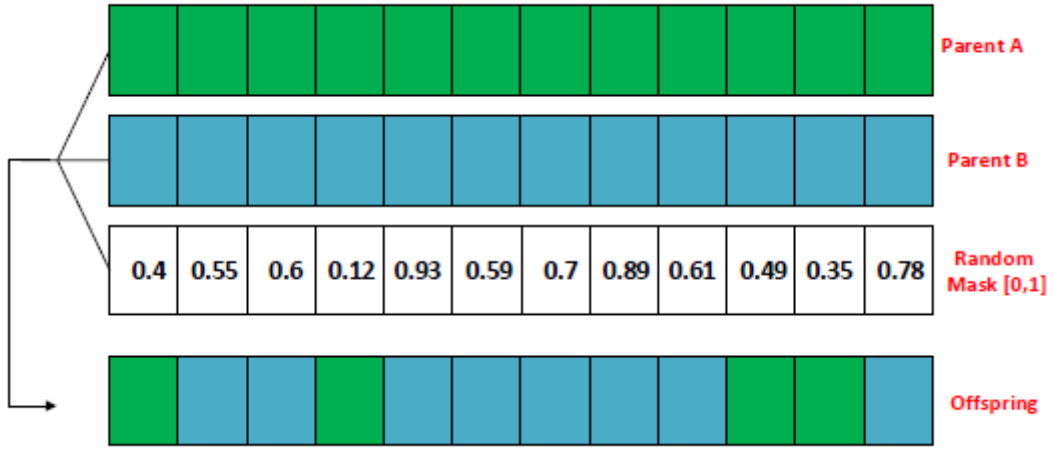
Bu yöntem, her ebeveynin uzunluğu boyunca iki rastgele çapraz nokta kullanır.Daha sonra iki nokta arasındaki genomlar değiştirilir.



Şekil 3.8. İki Noktalı Çaprazlama (S.S.Noorian, 2015).

-Üniforma Çaprazlama:

Üst öğeleri birkaç bölüme bölerek çalışan önceki operatörlerin aksine ve yavru üretmek için onları yeniden birleştirerek, Üniforma Çaprazlama, her geni bağımsız bir şekilde, göz önünde bulundurur. Her genin hangi ebeveynden miras alınacağına dair rastgele bir seçimdir. İlk olarak, 0 ile 1 arasındaki tekdüze bir dağılımdan bir N (ebeveynlerin boyutu) rastgele değişken dizisi oluşturulur. Her pozisyonda, rastgele değer tanımlanmış bir parametrenin (genellikle 0.5) altındaysa, gen ebeveyn A'dan; aksi takdirde ebeveyn B'den miras alınır.



Şekil 3.9. Üniforma Çaprazlama (S.S.Noorian, 2015)

-Sıralı Çaprazlama:

Sıralı çaprazlama, sıraya dayalı bir permütasyon problemimiz olduğunda kullanılır. Bu tür problemlerde, üretilen yavru, rekombinasyon dan sonra tekrarli genlere sahip olmamalıdır. Ek olarak, amaç, ikinci ebeveynden genlerin göreceli sırasını iletmektir. Örneğin, seyahat eden satıcı probleminde olduğu gibi[1], her şehir, başlangıç noktası dışında bir kereden fazla ziyaret edilemez.

Sıralı çaprazlama aşağıdaki gibi hareket eder:

1. Birinci ebeveyn boyunca iki rastgele çapraz nokta oluşturulur.
2. İki çapraz nokta arasındaki bölüm, birinci ebeveynden aynı pozisyondaki ilk yavruya kopyalanır.
3. İlk ebeveyn sol ve sağ bölümlerinden kalan kullanılmayan genler, ikinci ebeveyn sırasına göre ilk yavruya kopyalanır.

4- İkinci yavru da aynı şekilde ve ebeveynlerin rolleri tersine çevrilerek yaratılır.

-çaprazlama olasılığı:

[1] Traveling Salesman Problem

Çaprazlama olasılığı, seçilen ebeveyn üzerinde çaprazlama işlem gerçekleştirme olasılığını belirleyen bir parametredir. Bu parametre genellikle [0.5,1.0] aralığında belirlenir. Ebeveynleri seçtikten sonra, 0 ile 1 arasında rastgele bir sayı oluşturulur. Sayı seçilen ebeveynlerden daha düşükse, çaprazlama işlemi gerçekleştirilmez ve iki ebevey bir sonraki nesle kopyalanır; aksi takdirde, çaprazlama operatörü aracılığıyla iki yeni yavru oluşturulur(Eiben vd., 2003).

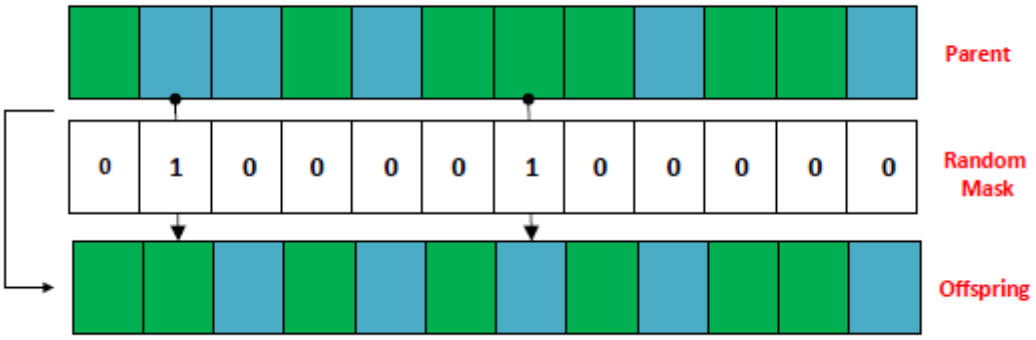
3.1.1.6.4.2. Mutasyon

Mutasyon, popülasyondaki genetik çeşitliliği koruyan basit bir arama operatörü olarak görülür. Bu operatör ebeveyn olarak yalnızca bir kromozom kullanır ve ebeveyn kromozomun yapısında birkaç rastgele değişiklik yaparak yavruları yaratır. Son bölümde tartıştığımız gibi, çaprazlama operatörü, daha iyi çözümleri çıkarmak için mevcut nesil çözümlerden yararlanmak için kullanılır. Ancak, mutasyon operatörü, algoritmanın global optimuma yakınsamasını sağlamak için tüm arama alanını keşfetmek için kullanılır (Sivanandam ve Deepa, 2008), (Eiben ve ark, 2003).

Mutasyon ayrıca algoritmanın local optimizasyonda takılıp kalmasını engellemeye çalışır. Kısaca, farklı mutasyon biçimlerinden bazıları şu şekilde sıralanmıştır (Sivanandam ve Deepa, 2008), (Eiben vd., 2003).

- Flip Mutasyonu:

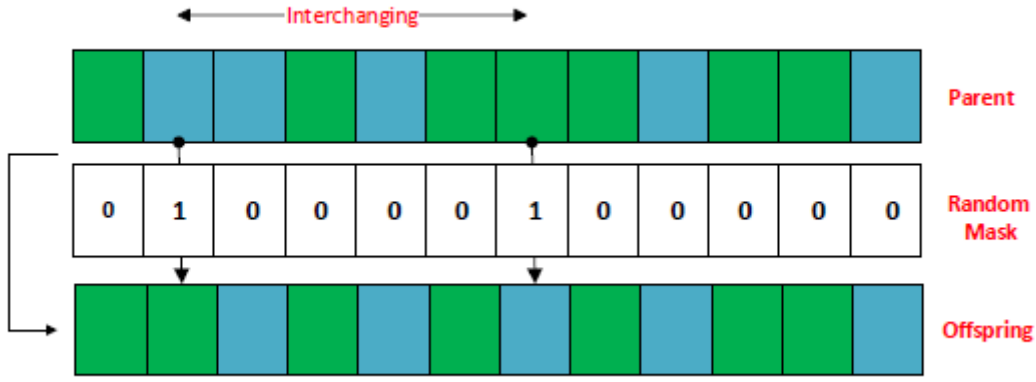
Bu mutasyon biçimi, genellikle binary çözümler gösterimi için kullanılır. Burada her gen ayrı ayrı ele alınır ve küçük bir olasılıkla çevrilebilir (0-1).



Şekil 3.10. Flip Mutasyonu (S.S.Noorian, 2015).

-Takas Mutasyonu:

Bu formda, kromozom içinden iki gen rastgele seçilir ve değerlerini değiştirir.



Şekil 3. 11. Takas Mutasyonu (S.S.Noorian, 2015).

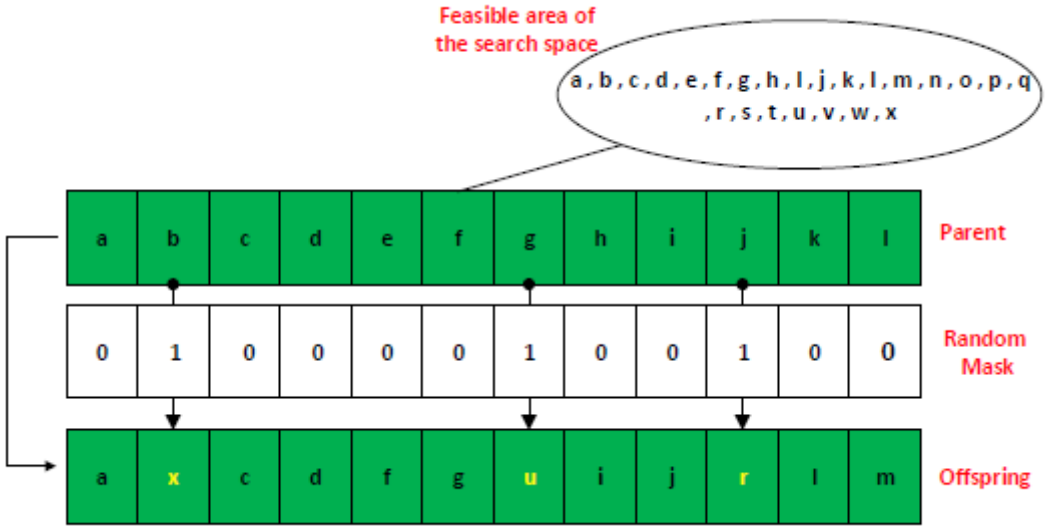
-İnversiyon Mutasyonu:

Bu operatör, kromozomun bir alt kümesini rastgele seçerek ve kromozomda göründükleri genomların sırasını tersine çevirerek çalışır.

Şekil 3. 12. İnversiyon Mutasyonu (S.S.Noorian, 2015)

-Üniforma Mutasyon:

Bu şema, ebeveyn kromozomun tüm genomlarını bağımsız olarak ele almak için tasarlanmıştır. Yinelemeli bir yöntemdir ki ilk genomdan başlar ve kromozomun tüm uzunluğu boyunca ilerler. Her gen için 0 ile 1 arasında rastgele bir sayı oluşturulur. Üretilen sayı, mutasyon olasılığından daha küçükse, arama alanının pratik alanından genom için yeni bir rastgele değer seçilebilir; aksi takdirde operatör, genomun değeri değişmez.



Şekil 3.13. Üniforma Mutasyon (S.S.Noorian, 2015).

3.1.1.6.4.3. Seçim

Seçim mekanizması basitçe, puana bağlı olarak bir sonraki toplumun ebeveyn adayları arasından bazı kişilerin seçilmesi olarak tanımlanır. Başka bir deyişle, ebeveyn topluluk bir seçim mekanizması ile seçilir. Böylelikle hangi bireylerin çaprazlama ve mutasyon işlevlerine maruz kaldığı belirlenir (Sivanandam ve Deepa, 2008).

Çeşitli Seçme yöntemleri aşağıdaki gibidir (Sivanandam ve Deepa, 2008), (Eiben vd., 2003):

-Rulet Çarkı Seçimi:

Rulet seçimi, birkaç bölüm bir rulet çarkı aracılığıyla doğrusal bir aramadır. Her bölüm, bireyin uygunluk değerleriyle orantılı olarak ağırlıklandırılır. Bu çok yaygın bir seçim yöntemidir, ancak doğru bireyler seçilmesini garanti etmez fakat doğru bireyler seçme şansını artırır. Ancak en iyi bireylerin uygunluk değerleri çok farklı olduğunda bu yöntemin bir problemi olacaktır. Örneğin, en iyi bireysel uygunluk % 95 ise rulet Çarkı Seçimi % 95'ini kaplar. Bu nedenle, diğer kromozomların seçilmek için çok az şansları vardır ve çok hızlı yakınsama ile sonuçlanır.

-Rastgele Seçim:

Bu, popülasyondan rastgele bir ebeveyn seçen stokastik bir yöntemdir. Rastgele seçim, uygunluğu hesaba katmaz.

-Turnuva seçimi:

En yaygın ve ideal seçim stratejisidir seçici basıncını ayarlayabilen ve nüfusun N bireyleri arasında bir rekabet düzenleyerek nüfus çeşitliliğini korur. Turnuva kazananı, en yüksek en yüksek uygunluk fonksiyonu sahip olan ve en iyi bireydir. Kazanan, yeni çocuğunuzu gelecek nesil için üreme süreciyle yetiştirecek ebeveyn olarak seçilir.

-Sıra seçimi:

Sıra seçimi önce popülasyonu sıralar ve ardından her kromozom bu sıralamadan uygunluk alır. En kötü durumda uygunluk 1, ikinci en kötü 2 vs... Ardından, tüm kromozomların, uygunluklerine göre seçilme şansıleri vardır.

3.1.1.6.4.4. Kullanım Kısıtlamaları

Kısıtlı optimizasyona GA'yı uygulamak için gerekli bir bileşen, kısıtlamaların nasıl ele alınacağıdır çünkü kromozomları manipüle etmek için kullanılan genetik operatörler olursuz (infeasible) yavrular verebili. GA ile kısıtlamaları ele almak için önerilen birkaç teknik vardır (Kim ve Myung,1996), (Orvosh ve Davis,1994). Michalewicz bu problemle ilgili bir anket yaptı (Michalewicz,1995), (Michalewicz vd., 1996) Mevcut teknikler şu şekilde sınıflandırılabilir

• Reddetme stratejisi

• Onarım stratejisi

•Cezalandırma stratejisi

Bu stratejilerin her birinin avantajları ve dezavantajları vardır.

3.1.1.6.4.4. 1.Reddetme Stratejisi

Reddetme stratejileri, arama sırasında yalnızca olurlu çözümlerin tutulduğu ve ardından olursuz çözümlerin otomatik olarak atıldığı basit bir yaklaşımı temsil eder. Bu tür stratejiler, arama uzayının olursuz çözümlerinin payı çok küçükse düşünülebilir(Talbi, E, G. 2009).

3.1.1.6.4.4. 2.Onarım Stratejisi

Bir kromozomu onarmak, olursuz bir kromozomu almak ve bazı onarım prosedürleriyle olurlu bir kromozom oluşturmak anlamına gelir. Pek çok kombinasyonel optimizasyon problemi için, bir onarım prosedürü oluşturmak nispeten kolaydır. Liepins ve çalışma arkadaşları, onarım stratejisinin gerçekten hem hız hem de performans açısından diğer stratejileri geride bıraktığını gösterdiler (Liepins vd., 1990), (Liepins ve Potter, 1991).

Onarım stratejisi, olursuz bir yavruyu olurlu bir yavruları dönüştürmek için deterministik onarım prosedürünün varlığına bağlıdır. Yöntemin zayıflığı, probleme bağlı olmasıdır. Her bir sorun için özel bir onarım algoritması tasarlanmalıdır. Ayrıca, bazı problemler için, olursuz kromozomları tamir etme süreci, orijinal problemi çözmek kadar karmaşık olabilir. Onarılan kromozom ya sadece değerlendirme için kullanılabilir ya da popülasyondaki orijinal kromozomun yerini alabilir (Gen vd., 2008).

3.1.1.6.4.4.3. Cezalandırma Stratejisi

Yukarıdaki bu stratejiler, hiçbir zaman olursuz çözümler üretmemeleri avantajına sahiptir, ancak olurlu uzayın dışında hiçbir noktayı dikkate almama dezavantajına sahiptir. Oldukça kısıtlı bir problem için, olursuz çözüm, popülasyonda nispeten büyük bir paya sahip olabilir. Böyle bir durumda, olurlu uzaydeki genetik aramayi onaylarsak, olurlu çözümler bulmak zor olabilir (Gen vd., 2008). Örneğin bizim çalışmamızda su kalitesi sensörlerinin çalışabilmesi için nehir suyuna yerleştirilmesi gerekmektedir. Sonuç olarak olurlu uzay nehir çevresi, olursuz uzay Dejirmandere havzasının nehir çevresinde olmayan diğer alanları, yasadışı uzay Dejirmandere havzası dışındaki alanlar denilmektedir.

3.1.1.6.4.4.Sonlandırma Koşulları

Algoritmayı durdurabilecek çeşitli koşullar vardır. Aşağıdakiler birkaç sonlandırma tekniğidir (Sivanandam ve Deepa, 2008):

-Geçen zaman:

Evrimsel süreç, önceden belirlenmiş bir süre geçtikten sonra sona erecektir.

-Nesil sayımı:

Belirli sayıda nesil geçtikten sonra evrimi sonlandırır.

-Durgunluk:

Bu durum popülasyonda en iyi iyileşme yoksa (uygunluk önceden belirlenmiş sayıda nesil içinde veya belirli bir zaman aralığında) bu durum sürecini durdurur.

-Hedef uygunluk:

Popülasyondaki en az bir aday belirtilen uygunluk puanına ulaştığında algoritma sonlandırılır.

-Kullanıcı İptali:

Bu koşul, kullanıcıların istedikleri zaman GA'yı sonlandırmalarına izin vermek için uygulanır.

3.2. Optimization

Optimizasyon algoritmaları, bir veya daha fazla hedefi maksimize etmek veya minimize etmek için olası tüm aday çözümler arasından optimal çözümü bulmak için tasarlanmış güçlü arama algoritmalarıdır (Mirjalili ve Dong, 2020). Optimizasyon algoritmaları deterministik ve stokastik olmak üzere iki gruba ayrılabilir. Deterministik yöntemler matematiğe dayanır ve gradyan arama yöntemleri veya ikinci dereceden yöntemler olabilir (Du ve Swamy, 2016). En deterministik yöntemler, elde edilen çözümlerin kalitesiyle değil, yalnızca problemin talimatlarıyla (yani problem tanımıyla) ilgilenen, bilgisiz yöntemlerdir. Bu nedenle, hedefe olan mesafeleri hakkında hiçbir bilgileri yoktur. Deterministic yöntemler, en iyi çözümü bulmayı garanti eden kesin algoritmalardır. Bu tür algoritmaların her adımı eşit derecede iyidir. Ancak problemin boyutu arttıkça hesaplama yetenekleri azalır ve yavaşlar. "breadth-first" arama ve "brute-force" arama yöntemleri, kör arama yöntemleri olarak da bilinen deterministik yöntemlerin en popüler örnekleri arasındadır (Mirjalili ve Dong, 2020). Deterministik yöntemlerden farklı olarak, stokastik yöntemler makul bir çözüme ulaşmak için çeşitli rastgele kurallardan yararlanır (Yang, 2011). Stokastik algoritmalar herhangi bir türetmeye ihtiyaç duymazlar ve gradyan tabanlı da değille. Bu yöntemlerin çoğu bilgili yöntemlerdir. Deterministik yöntemlerden farklı olarak, stokastik arama yöntemleri, hedefe olan yaklaşık mesafeyi gösteren ek bilgileri kullanır. Bu tür algoritmalara genellikle heuristic arama algoritmaları denir (Kanal ve Kumar, 2012). Heuristic algoritmalar, bilinçli olarak

arama uzayının farklı bölümlerini keşfederek, bilinçli kararlar vererek, deterministik yöntemlerin aksine, her seferinde daha iyi veya daha kötü adımlar atmalarını sağlar (Mirjalili ve diğerleri, 2020). Tüm arama uzayını aramak yerine, bu algoritmalar bilinçli bir aramaya izin verirler, hesaplama yükünü büyük ölçüde azaltır ve sorunun boyutu ne olursa olsun daha hızlı sonuçlar verirler. Bu, elbette, onları oldukça popüler hale getirdi. Ancak, heuristic arama algoritmaları en iyi sonuçları garanti etmezler. Bunun yerine, kısa sürede en iyiye en yakın makul bir sonuç verirler. "Greedy" arama ve "hill-climbing" algoritmaları, heuristic algoritmaların en popüler örneklerinden bazılarıdır. Literatür ayrıca hibrit algoritmaları da bildirir, yani hem deterministik hem de stokastik özellikler gösterebilirler. Örneğin daha önce stokastik bir yöntem olarak sınıflandırılan "'hill-climbing" algoritması, aynı başlangıç noktasından başlatılırsa her seferinde aynı sonucu verecektir. Ancak başlangıç noktasında rastgelelik varsa nihai sonuç değişebilir. Ancak hibrit yöntemlerde rastgele bir bileşen olduğu için bu yöntemler optimizasyon literatüründe sıklıkla stokastik algoritmalar olarak sınıflandırılmaktadır (Yang, 2011). Özetlemek gerekirse, heuristic algoritmalar makul bir zaman çerçevesinde iyi performans gösterebilir. Bunun nedeni, algoritmanın bir adım atmadan önce en makul seçeneği seçebilmesidir. Ayrıca, bu şekilde eğitimli kararlar verme yeteneği, arama alanının boyutunu önemli ölçüde azaltır. Ancak bu yöntemler mümkün olan en iyi sonucu bulmada güvenilir kabul edilemez (Mirjalili ve Dong, 2020). Ayrıca, heuristic algoritmalar probleme özeldir ve bu nedenle probleme özel bilgiye ihtiyaç duyar. Örneğin, "best-first arama" algoritması, belirli bir kurala göre arama uzayında bulunan tüm konumlar arasından en 'umut verici' konumu seçer (Burns ve diğerleri, 2010). Probleme bağlı heuristic algoritmalar, araştırmacıları farklı problemlere kolayca uygulanabilen metaheuristic algoritmalar geliştirmeye motive etmiştir. Metaheuristic algoritmalar, tamamen adapte edilmek zorunda kalmadan birçok zor optimizasyon problemi için optimum çözümler sunabilmektedir. Metaheuristic kelimedeki Yunanca kökenli 'meta' öneki, probleme özel heuristic algoritmaların aksine, bu algoritmaların yüksek seviyeli heuristic algoritmalar olduğuna işaret etmektedir (Boussaïd ve diğerleri, 2013). Metaheuristic yöntemler, heuristic yöntemlere göre arama alanını daha aktif ve verimli bir şekilde keşfetme kapasitesine sahiptir. Metaheuristic algoritmaların bir diğer avantajı da eğim tabanlı algoritmalar olmamasıdır. Böylece türev hesaplamalarına gerek kalmadan optimal sonucu verebilirler. Bu sayede büyük boyutlu ve karmaşık yapıdaki problemler kısa sürede başarılı bir şekilde çözülebilmektedir (Talbi, 2009). Metaheuristic algoritmalar, çok önemli iki işleç, kullanma ve keşif kullanır.

Kullanma, arama uzayındaki en iyi çözümler etrafında yerel bir arama yürütürken, keşif, arama uzayının farklı bölgelerini araştırır. "hill-climbing" algoritmasında, bir başlangıç noktasından başladıktan sonra algoritma her seferinde en iyi komşuyu seçerek ilerler. Karmaşık modellerde (yani, global minimum veya maksimum ile birlikte local optimumu içeren modeller), yalnızca bir komşuluk araması (yani kullanma) yürüten yöntem, arama alanındaki local maksimum veya minimumlardan birinde takılıp kalır. Bunun nedeni, algoritmanın her adımda en yakın komşularından yalnızca en yüksek veya en düşük olanı seçmesidir (Mirjalili ve Dong, 2020). Keşif ve kullanma arasındaki uzlaşma, metaheuristic arama algoritmalarının başarısını büyük ölçüde etkiler. Örneğin bir algoritma sadece keşif operatörünü kullanıyorsa yani global olarak arama yapıyorsa muhtemelen maksimum zirvede bir nokta bulacaktır ancak bu nokta beklenen kalitede olmayacaktır çünkü algoritma bu noktayı küçük hamleler yaparak iyileştiremez. Buna karşılık, yalnızca kullanma operatörünü kullanan bir algoritma, yerel maksimum tuzaklardan birinde çok hızlı bir şekilde yakınsayacaktır. Bu nedenle, problem ne olursa olsun, global optimum noktaya ulaşmak için bu iki operatör arasında uygun bir denge şarttır. Keşif ve kullanma arasındaki uzlaşma, metaheuristic arama algoritmalarının başarısını büyük ölçüde etkiler. Örneğin bir algoritma sadece keşif operatörünü kullanıyorsa yani global olarak arama yapıyorsa muhtemelen maksimum zirvede bir nokta bulacaktır ancak bu nokta beklenen kalitede olmayacaktır çünkü algoritma bu noktayı küçük hamleler yaparak iyileştiremez. Buna karşılık, yalnızca kullanma operatörünü kullanan bir algoritma, yerel maksimum tuzaklardan birinde çok hızlı bir şekilde yakınsayacaktır. Bu nedenle, problem ne olursa olsun, global optimum noktaya ulaşmak için bu iki operatör arasında uygun bir denge şarttır.

3.2.1. Tek Amaçlı Optimizasyon

Optimizasyon algoritmaları, bir amaç fonksiyonunu maksimize etmek veya minimize etmek için problem için optimum parametre değerlerine ulaşmayı amaçlar. Optimizasyon algoritmaları, bir amaç fonksiyonunu maksimize etmek veya minimize etmek için problem için optimum parametre değerlerine ulaşmayı amaçlar. Literatürde amaç fonksiyonu genellikle uygunluk veya maliyet fonksiyonu olarak adlandırılır. Tek amaçlı optimizasyon algoritmaları, tek bir amaç fonksiyonunu minimize veya maksimize etmeye amaçlar.

Genel olarak, amacı minimize etmek için tek amaçlı bir optimizasyon problemi matematiksel olarak şu şekilde ifade edilir (Yang, 2011):

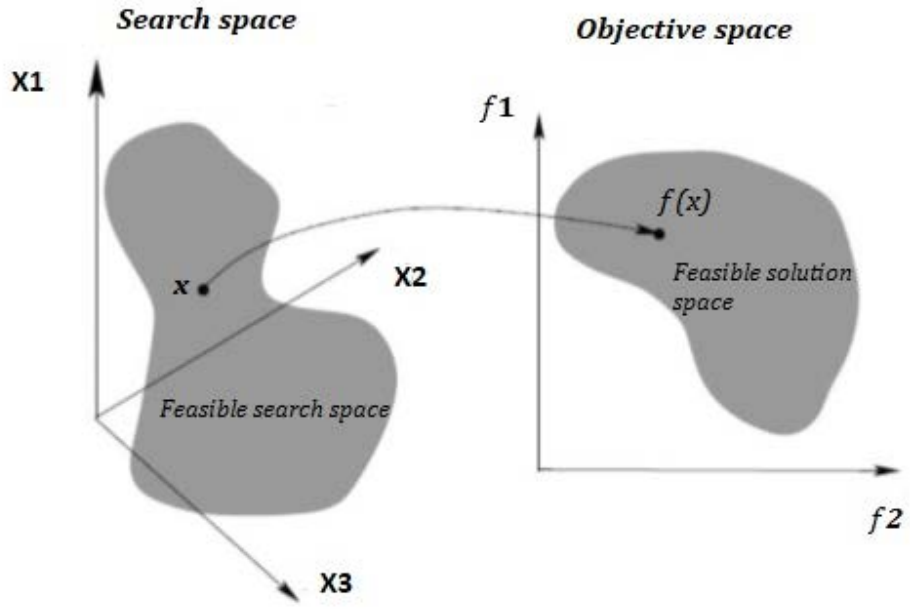
$$\min_{x \in \Re n} \quad f(x_1, x_2, \ldots, x_n) \tag{3.1}$$

$$g_i (x_1, x_2, \ldots, x_n) \geq 0, \quad i=1,2, \ldots, l \tag{3.2}$$

$$h_i (x_1, x_2, \ldots, x_n) = 0, \quad i=1,2, \ldots, p \tag{3.3}$$

$$lb_i \leq x_i \leq ub_i, \quad i=1,2, \ldots, n \tag{3.4}$$

"f" minimize edilecek amaç fonksiyonunu ifade eder, "n" boyutlu uzayda herhangi bir gerçek değeri alabilen "x", tasarım veya karar değişkeni olarak adlandırılır. Karar değişkenleri tarafından oluşturulan uzaya arama uzayı, amaç fonksiyonundan elde edilen değerleri içeren uzaya ise çözüm ya da amaç uzayı denir. Bazı optimizasyon problemlerinin arama uzayında kısıtlamaları olabilir. Bu kısıtlamalar aslında bir optimizasyon algoritmasının kaçınması gereken arama alanındaki boşluklardır. Arama uzayında problemin kısıtlarını karşılayan tüm olası karar değişkenlerini içeren bölgeye uygun arama uzayı, çözüm uzayında bu bölgeye karşılık gelen bölgeye uygun çözüm uzayı denir.



Şekil 3.14. Karar alanını ve ilgili hedef alanını gösterimi(Deb, 2001).

Şekil 3.14'de, uygun arama ve çözüm uzayı ile birlikte üç elemanlı karar değişkeni vektörünün iki amaçlı bir optimizasyon problemi için arama uzayı ve çözüm uzayı gösterilmeştir. Denklemler 2 ve 3, sırasıyla eşitsizlik "g_i" ve eşitlik "h_i" kısıtlama fonksiyonlarını tanımlar. Burada "l" eşitsizlik kısıtlamalarının sayısını, "p" eşitlik kısıtlamalarının sayısını, "lb_i" i-inci değişkenin alt sınırını ve "ub_i" i-inci değişkenin üst sınırını ifade etmektedir. $l = p = 0$ ise, optimizasyon problemi kısıtsızdır. $p \geq 1$, $l = 0$ ise problem eşitlik kısıtlıdır ve $l \geq 1$, $p = 0$ ise problemin eşitsizlik kısıtlı optimizasyon problemi olduğu söylenir (Yang, 2011).

3.2.2. Çok Amaçlı Optimizasyon

Çok amaçlı optimizasyon (MOO) probleminde, optimize edilmesi gereken birden çok amaç fonksiyonu vardır. Tek amaçlı optimizasyon ile karşılaştırıldığında, MOO'nun uygulanması çok daha zordur, çünkü optimal değerler çoklu ve çoğu zaman çelişen hedeflere hitap ederek aranır. MOO problemi, amaçları minimize etmek için aşağıdaki gibi matematiksel olarak formüle edilmiştir (Mirjalili ve Dong, 2020):

$$\min_{x \in \Re n} \quad f(X) = [f_1(X), f_2(X), \ldots, f_m(X)], \qquad m \geq 2 \tag{3.5}$$

$$X = (x_1, x_2, \ldots, x_n)^T \tag{3.6}$$

Burada, "(X)" amaç fonksiyon vektörüdür ve "m", amaç fonksiyonlarının toplam sayısını gösterir. Denklem 6, uygulanabilir çözüm uzayında "n" elemanlarının "X" karar değişkeni vektörünü ifade eder. Tek amaçlı optimizasyonda kullanılan kısıt denklemleri (Denklem 3.2 ve 3.3) ve sınır değer denklemi (Denklem 3.4) MOO için de geçerlidir.

Tek amaçlı optimizasyonda, çözüm uzayında elde edilen sonuçlar matematiksel büyüklük veya küçüklük operatörleri ile değerlendirilebilirken, MOO'da bu operatörler yetersiz kalmaktadır. Örneğin, bir MOO'da, bir minimizasyon problemi için iki farklı amaç fonksiyonu olduğunda, bir karar değişkeni vektörü bir amaç için minimum değeri verebilir, ancak diğer amaç için veremez. Yani optimize edilecek amaç fonksiyonlarının her ikisi de aynı noktada (aynı karar değişkeni ile) optimum değeri vermeyebilir, bu durumda tek bir

optimum değerden bahsedilemez. Bu nedenle, karar değişkenlerinin vektörlerinin birbirleriyle karşılaştırılabileceği bir operatöre ihtiyaç vardır. Bu operatör, en iyi karar değişkeni vektörlerini belirleyebilen Pareto optimal baskınlığıdır (Censor, 1977; Mirjalili ve Dong, 2020). Bu operatöre göre, bir karar değişkeni vektörü, ancak tüm amaç fonksiyonu değerlerinde diğeri kadar iyi ise ve en az bir amaç fonksiyonu değerinde diğerinden daha iyi ise diğerine baskındır. Pareto baskınlığı minimizasyon problemi matematiksel olarak şu şekilde ifade edilebilir (Deb, 2001; Tran vd., 2016);

Karar değişkeni vektörü $X_1(x_{1.1}, x_{1.2},\ldots, x_{1.n})^T$ diğer karar değişkeni vektörüne $X_2(x_{2.1}, x_{2.2}, \ldots, x_{2.n})^T$ hakimdir, aşağıdaki iki koşul karşılanırsa*:*

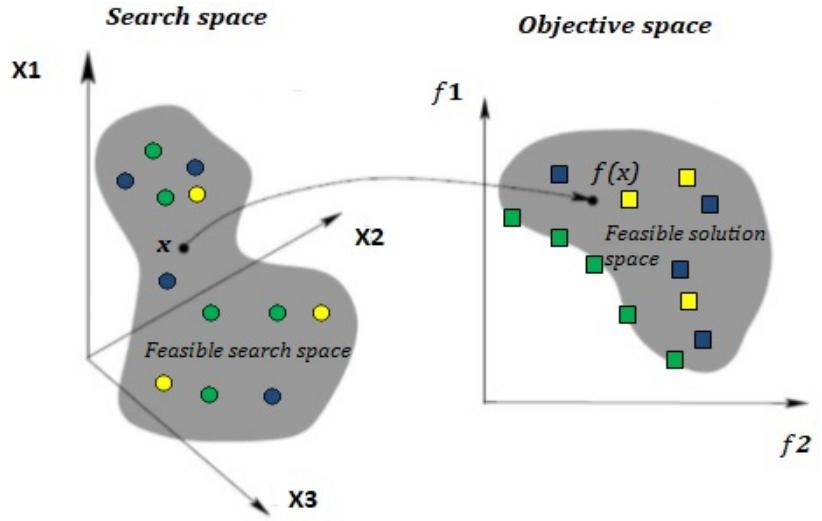
1- $\forall i \in (1,2, \ldots, m)$: $f_i\,(X_1) \leq f_i\,(X_2)$. (3.7)

Karar değişkeni vektörü X_1 her bir amaç fonksiyonunda X_2 kadar iyi bir değer vermelidir.

2- $\exists i \in (1,2, \ldots, m)$: $f_j\,(X_1) < f_j\,(X_2)$. (3.8)

Karar değişkeni vektörü X_1, en az bir amaç fonksiyonunda kesinlikle X_2'den daha iyi performans göstermelidir.

Herhangi bir karar değişkeni tarafından baskin edilemeyen karar değişkenleri kümesine Pareto optimal çözüm kümesi denir. Bu küme, amaç fonksiyonları arasında en iyi dengeyi sağlayan karar değişkenlerinin vektörlerinden oluşur. Pareto optimal çözüm kümesinin amaç fonksiyonlarında değerlendirilmesi ile elde edilen çözüm uzayındaki değerlerden oluşan kümeye Pareto cephesi denir. MOO'da oldukça önemli olan bu iki küme Şekil 3.15'te gösterilmektedir.



Şekil 3.15. Pareto optimal çözüm kümesi ve Pareto cephesi , after (Deb, 2001).

Şekil 3.15'de arama uzayındaki daireler üç boyutlu karar değişkeni vektörlerini temsil ederken, kareler iki boyutlu çözüm uzayındaki tepkilerini yani bu karar değişkeninin vektörleri fonksiyonlara konulduğundaki amaç değerlerini göstermektedir. $f1$ ve $f2$. Çözüm uzayında yeşil karelerle gösterilen küme Pareto cephesini oluşturur. Başka bir deyişle, yeşil kümeye başka kareler baskin olamaz.

Literatür, MOO problemlerini çözmek için iki genel yöntem bildirmektedir. İlk yöntem, basitliği nedeniyle literatürde sıklıkla kullanılan ağırlıklı toplam MOO yöntemidir (Marler ve Arora, 2010). Bu yöntemde MOO problemleri sabit ağırlıklı lineer fonksiyonlara dönüştürülür ve tek amaçlı optimizasyon problemleri gibi çözülür. Herhangi bir minimizasyon problemi için bu prosedürün matematiksel ifadesi aşağıdaki gibidir (Yang, 2014);

$$\min_{x \in \Re n} \quad f(X) = w_1 f_1(x) + w_2 f_{2+} \ldots w_m f_m(x), \qquad m \geq 2 \tag{3.9}$$

$$\sum_{i=1}^{m} Wi = 1 \qquad W_i > 0 \tag{3.10}$$

Denklem 3.9 ve 3.10'da görüldüğü gibi her bir amaç fonksiyonunun başında karar verici tarafından belirlenmesi gereken ağırlıklar vardır ve bu ağırlıkların toplamı 1 olmalıdır. Kısıt denklemleri (yani Denklem 2 ve 3). ve daha önce verilen sınır değer denklemi (yani Denklem 4), ağırlıklı toplam çok amaçlı optimizasyon yöntemi için de geçerlidir. Bu yöntemin anlaşılmasının kolay olması ve hesaplama yükünün olmaması çok avantajlıdır. Çünkü bu yöntem tek amaçlı optimizasyon gibi çözümü bulur ve baskın olmayan çözümlere ihtiyaç duymaz (Jin vd., 2001; Yang, 2014). İkinci yöntem, baskın olmayan çözümleri seçmek için Pareto kavramını (Denklem 3.7 ve 3.8) kullanır. Bu yöntemler tüm amaç fonksiyonlarını aynı anda optimize etmeyi amaçlar (Mirjalili ve Dong, 2020).

3.2.3. Evrimsel Algoritmalarla Çok Amaçlı Optimizasyon

Çok amaçlı optimizasyon amaçları için evrimsel algoritmaların kullanımı son yıllarda çok yaygın hale geldi. 1960'larda, Almanya ve Amerika Birleşik Devletleri'nde üç bağımsız kavramsal ve hesaplamalı gelişme gerçekleşti. Bu eğilim, bilimde evrimsel algoritmalar adı verilen yeni bir alanın ortaya çıkmasına yol açtı: Evrimsel stratejiler[1] (Rechenberg, 1965), Evrimsel programlama[2] (Fogel, 1966) ve GA'lar (Holland,1962). 1980'lerde bu gruplar arasında dinamik bir etkileşim ortaya çıktı ve bu, genetik programlama gibi alanda yeni dalların ortaya çıkmasına neden oldu.

Evrimsel algoritmanın çok amaçlı optimizasyon problemlerini çözme yeteneğinin ilk olarak 1960 yılında Rechenberg tarafından tanıtıldığı söylenebilir. Çok Amaçlı Evrimsel Algoritmanın[3] (MOEA) ilk gerçek uygulaması 1980'lerin ortasında gerçekleşti. O zamandan beri bu alanda çok fazla araştırma yapıldı ve bugün Çok Amaçlı Evrimsel Algoritma olarak biliniyor.

Çok amaçlı optimizasyonda, birkaç olası çelişkili amaç fonksiyonu için uğraşıyoruz ve genellikle bir fonksiyonun optimal çözümü, diğer amaç fonksiyonlar için en uygun çözüm olmayacaktır. Böyle bir senaryoda, optimal bir cevap yerine, Pareto optimal

[1] Evolutionary strategies
[2] Evolutionary programming
[3] Multiobjective Evolutionary Algorithm

çözümleri[1] adı verilen bir dizi optimal cevap olacaktır. Çeşitli metinlerde, çok amaçlı evrimsel optimizasyonun amacı, uygun dağıtım ile bir dizi Pareto optimal çözümü bulmaktır (Ehrgott ve Wiecek, 2005).

Tüm evrimsel yöntemler aşağıdaki hedefleri takip eder (Zitzler,1999):

1) Optimal Parto setine doğru ilerlemek için baskın olmayan çözümlere vurgu
2) Çözümler arasında uygun çeşitliliği elde etmek için dağınık çözümlere vurgu
3) Optimal Parto setine doğru daha hızlı ilerlemek için seçilmiş ve seçkin çözümlere vurgu

Çok amaçlı evrimsel algoritmalarıyla ilgili sorunlardan biri, çok sayıda amac fonksiyonu olan sorunlarda, nüfusun birçok üyesinin baskın olmayan çözüm kümesine ait olmasıdır. Açıktır ki, tüm cevaplar seçilip sonraki nesillere aktarılmaz. Bu tür sorunlar, evrimsel algoritmalarda elitizm için yöntemlerin tanımlanmasına yol açtı (De Jong, 1975).

Elit çözüm, bir toplulukta puanı en iyi olan çözüm demektir. Elitizm ya da en iyi çözüm/çözümlerin saklanması ve bir sonraki nesile eklenmesinin GA'nın başarımını önemli ölçüde etkilediği görülmüştür (Zitzler, 1999). Ancak elitizmin dikkatli bir biçimde uygulanması gerekir. Eğer elitizm kontrollü bir biçimde uygulanmazsa çeşitlilik kaybı söz konusu olabilir.

NSGA-II, GA'nın en yaygın olarak kullanılan MOO modlarından biridir ve Bu çalışmada aşağıda belirtilen sebeplerden dolayı NSGA-II algoritmayı seçtik:

- Bu, sıralamada diğer yöntemlere göre daha hızlı bir çözümdür ve önceki algoritmaların hesaplama karmaşıklığını azaltmıştır (Bui & alam, 2008: 58-63).

- Diğer algoritmalardan daha tek tip çözüm cephesi elde etmek ve cevapların etrafındaki noktaların yoğunluğunu tahmin etmek için kalabalık mesafesini kullanır. Unutulmamalıdır ki kalabalık mesafesi tek cepheye saçılma açısından çözümlerin daha iyi seçilmesinde kullanılan bir faktördür (Masumi vd., 2010: 6)

Aşağıda, NSGA-II yöntemi açıklıyoruz.

[1] Pareto optimal solutions

3.2.3.1. Bastırılamayan Sıralamalı Genetik Algoritma-II (NSGA -II)

NSGA-II prosedürü (Dep vd., 2002), popüler olarak kullanılan Evrimsel Çok Amaçlı Optimizasyon (EMO) prosedürlerinden biridir ve aşağıdaki üç özelliğe sahiptir:

1. Elitist bir ilke kullanır,

   Mevcut neslin en iyi kromozomlarını seçip bir sonraki nesile aktarmak, elitist seçim olarak bilinir. Bu strateji, GA tarafından elde edilen çözüm kalitesinin bir nesilden diğerine düşmeyeceğini garanti eder.

2. Çeşitliliği korumak için açık bir mekanizma kullanır

   Bu algoritma, Pareto Cephesi boyunca çözüm çeşitliliğini korumak için bir kalabalık-mesafe hesaplaması kullanır.

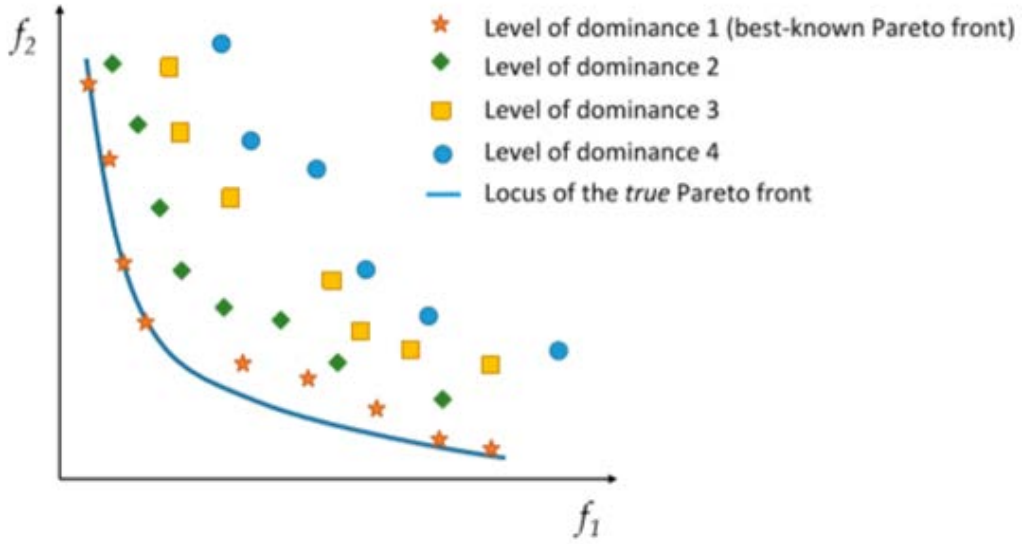
3. baskin olmayan çözümlere vurgu yapar.

NSGAII'de, her bir çözüm için Pareto sıralaması oluşturmak için baskin olmayan bir sıralama yaklaşımı kullanılır ve yoğunluk tahminini uygulamak[1] için bir kalabalıklaşma mesafesi atama yöntemi[2] uygulanır. İki çözüm arasındaki uyum atamasında (fitness assignment) NSGA II, daha düşük sıra değerine sahip (daha düşük cephe değerine sahip) noktayı veya her iki nokta da aynı cepheye aitse daha çok kalabalık mesafe değerine sahip noktayı tercih eder.

3.2.3.1.1. Pareto Sıralaması

NSGA-II, her nesil için çözümleri baskın olmama durumuna göre sınıflandırır ve birkaç cephe, sıra veya baskınlık düzeyi oluşturarak çözümleri sıralar. İlk cephe, baskın olmayan tüm çözümlerden oluşur. İkinci cephe, sadece bir çözümün baskın olduğu çözümlere sahiptir. Üçüncü cephe, diğer iki çözümün baskın olduğu çözümlere sahiptir ve cepheler, tüm çözümler sınıflandırılana kadar oluşturulur. Sıralanmış bir NSGA-II çözüm seti Şekil 3.16'de gösterilmiştir.

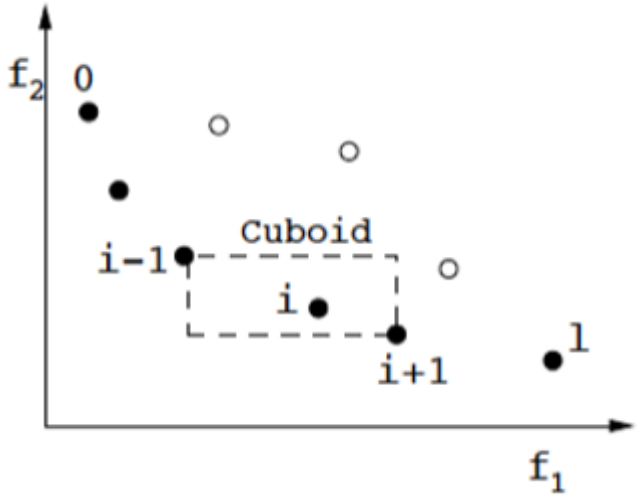
[1] Implement density estimation
[2] crowding distance assignment method



Şekil 3.16. Baskınlık düzeyleri. f 1 ve f 2 fonksiyonları minimize edilmiştir (Chicco ve Mazza, 2020)

3.2.3.1.2. Yoğunluk Tahmini ve Kalabalık Mesafesi

Popülasyondaki belirli bir çözümü çevreleyen çözüm yoğunluğunun bir tahminini elde etmek için, çözümlerin her biri boyunca bu noktanın her iki tarafındaki iki noktanın ortalama mesafesini hesaplıyoruz. Bu $i_{distance}$ miktarı, köşeler olarak en yakın komşular kullanılarak oluşturulan kübik çevrenin bir tahmini olarak hizmet eder (buna kalabalık mesafesi deyin). Şekil 3.17'de, önündeki *i* çözümünün kalabalık mesafesi (içi dolu dairelerle işaretlenmiştir), küboidin (kesikli bir kutu ile gösterilmiştir) ortalama yan uzunluğudur.



Şekil 3.17. Kalabalık mesafe hesabı. *İ*çi dolu dairelerle işaretlenen noktalar, aynı dairelerle işaretlenen noktalar, aynı baskın olmayan cephenin çözümleridir (Deb vd., 2002)

Kalabalık-mesafe hesaplaması, çözümler her bir amaç fonksiyon değerine göre artan büyüklük sırasına göre sıralanmasını gerektirir. Bundan sonra, her bir amaç fonksiyon için sınır çözümlerine (en küçük ve en büyük fonksiyon değerlerine sahip çözümler) sonsuz bir mesafe değeri atanır. Diğer tüm ara çözümlere, iki bitişik çözümün fonksiyon değerlerindeki mutlak normalleştirilmiş farka eşit bir mesafe değeri atanır. Bu hesaplama diğer amaç fonksiyonlarıyla devam ettirilir. Genel kalabalık mesafe diğeri, her bir amaca karşılık gelen ayrı mesafe değerlerinin toplamı olarak hesaplanır. Her bir amacın işlevi, kalabalık mesafesi hesaplanmadan önce normalleştirilir. Tablo 3.1'da gösterildiği gibi algoritma, baskın olmayan bir set *I*'deki (aynı cephede) tüm çözümlerin kalabalıklaşma mesafesi hesaplama prosedürünü ana hatlarıyla belirtir.

Tablo 3.1. Kalabalık mesafe ataması için pseudocodu

Crowding-distance-assignment (I)

$L = \|I\|$	number of solutions in "***I***"
For each i, set $I[i]_{distance} = 0$	initialize distance
For each objective m	
$I = sort\ (I, m)$	sort using each objective value
$I[1]_{distance} = I[L]_{distance} = \infty$	so that boundary points are always selected
For i = 2 to (L-1)	for all other points
$I[i]_{distance} = I[i]_{distance} + (I[i+1] * m - I[i-1] * m) / (\int_m^{max} - \int_m^{min})$	

Burada, I [i] $.m$, I kümesindeki (I cebhesindeki) m'inci amaç fonksiyon değerini ifade eder ve $\int_m^{max} ve\ \int_m^{min} parametreleri$, m'inci amaç fonksiyonunun maksimum ve minimum değerleridir (Deb vd., 2002).

3.2.3. 2. NSGA-II Algoritmasının Çalışma Yöntemi

NSGA-II algoritması ilk olarak ilk popülasyonu (kromozomlar veya çözümler) rastgele seçerek başlar ve bu çözümlerin fitness fonksiyonu tarafından değerlendirilir ve uygunluk puanı bu çözümlerin her birine atanır (Şekil 3.18 'nın 2 ve 3 Satırları). Bir sonraki adımda, NSGA-II algoritması, baskın olmama durumunu göz önünde bulundurarak, birkaç cephe veya sıra oluşturarak çözümleri sıralar. Daha sonra ebeveynlerin çözümlerinden yavru çözümler yaratılır. Sonra, NSGA-II'nin sonlandırma koşulu incelenir (Satır 6). Bu koşul sağlanırsa NSGA-II sona erer, NSGA-II'nin son koşulu karşılanmazsa NSGA-II devam eder ve her nesil için NSGA-II, baskın olmama durumunu göz önünde bulundurarak, birkaç cephe veya sıra oluşturarak bireyleri ebeveyn ve yavru popülasyonlarından sıralir (Satır 8 ve 9). İlk cephe, baskın olmayan tüm çözümlerden

oluşur. İkinci cephe, sadece bir çözümün baskın olduğu çözümlere sahiptir. Üçüncü cephe, diğer iki çözümün baskın olduğu çözümlere sahiptir ve cepheler, tüm çözümler sınıflandırılana kadar oluşturulur. Aynı cephenin çözümleri için, çözüm çeşitliliğini korumak için kalabalık mesafesi kullanılarak başka bir sıralama yapılır (Satır 10). Hesaplamadan sonra çözümler, kalabalık mesafesine göre azalan şekilde sıralanır. Her iki sıralama prosedürü, cephe ve kalabalık mesafesi, seçim operatörü tarafından kullanılır (Satır 11). Bu operatör, yeni nesil oluşturmak için alt cephe veya sıra çözümlerini seçer; ve alt sıradaki cephede yeterli sayıda çözüm bulunmaması nedeniyle, üst sıradaki cephede yalnızca bazı çözümlerin seçilmesi gerekiyorsa, daha büyük bir kalabalık mesafesine sahip çözümler seçilir (Satır 14). Daha sonra bu seçilen çözümler, öncekilerden yeni çözümler yaratmak için mutasyon ve çaprazlama operatörüne tabi tutulur. Bu işlem NSGA-II algoritmasının bitiş koşuluna kadar tekrarlanır. Son olarak, NSGA-II algoritmasının son neslindeki ilk cephe, Pareto optimal çözümler olarak seçilir (Deb ve diğerleri 2002). NSGA-II'nin sözde kodu, Şekil 3.18'de gösterilmiştir.

NSGA II Algorithm

Determine objective function

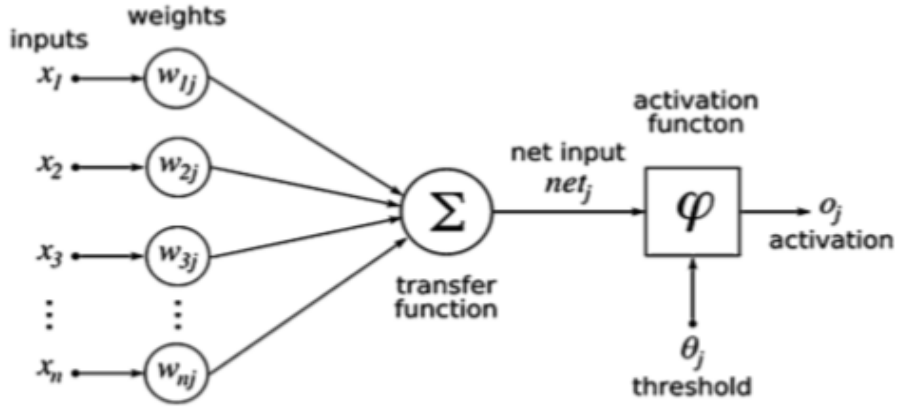
1. initialize population: the number of population;
2. Create an initial population S
3. Evaluate objectives values
4. Assign Rank (level) Based on Pareto dominance - sort
5. Generate **offspring** population for size S
6. for i = 1 : Max generation **do**
7. **for each parent and offspring** ∈ S **do**
8. Assign Rank (level) Based on Pareto dominance - sort
9. Generate sets of nondominated **solutions**
10. **Determine Crowding distance**
11. Crossover and mutation
12. Loop based on existing solution to next generation
13. **end for**
14. Select population on the lower front (lower rank) with high crowding distance
15. Generate next generation
16. **end for**

Output : The best individuals found during the evolution

Şekil 3.18. NSGA-II Algoritmasının pseudocodu (LI, Zhi, vd., 2019).

3.3. Yapay Sinir Ağı[1]

Yapay sinir ağları (YSA), insan beyninin sinirsel yapısından ilham alan bilgi işleme modelleridir. YSA paralel dağıtılmış bir yapıya sahiptir ve ağ üzerinden bilgi besleyerek girdilerden değerleri hesaplayabilen "nöronlar" adı verilen bir dizi işlem elemanından oluşur (Anderson ve McNeill, 1992). YSA, mimarileri, transfer işlevleri ve eğitim algoritmaları veya öğrenme kuralları ile karakterize edilir. YSA, Şekil 3.22'de gösterildiği gibi üç katmandan, verileri alan bir girdi katmanından (x_i), girdi ve çıktı katmanları arasında bilgiyi işleyen bir veya daha fazla gizli katmandan ve hesaplanan bilgiyi gönderen bir çıktı katmanından (O_J) oluşur. Şekil 3.19, en yaygın ağlardan birini, gizli bir katmana sahip katmanlı ileri beslemeli yapay sinir ağını göstermektedir. Bu ağlardaki katmanlar, aralarından geçen bilgilerin etkisini belirleyen ağırlıklarla (uyarlanabilir sinaptik ağırlık, wij, burada "I" katman indeksi ve "j" nöron indeksidir) ilişkili iletişim bağlantıları ile birbirine bağlıdır. Bu toplamın sonucu, "netj", daha sonra, bu toplam belirli bir eşiği veya "θ_J" sapmasını aşarsa, bir nöronun "O_j" çıkışını üreten bir transfer fonksiyonu "φ" tarafından dönüştürülür. Ağırlık değerleri deneyim veya çıktı doğrulaması ile belirlenir (EHSAN, R., 2017).



Şekil 3.19. Yapay sinir ağı şeması (EHSAN, R., 2017).

[1] Artıficıal neural networks

YSA çalışma süreci iki aşamaya ayrılır: İlk aşama, dış ortamdaki değişikliklere uyarlanmış öğrenme yoluyla bağlantı ağırlıklarının düzeltildiği eğitim olarak adlandırılır. Yapay sinir ağlarında eğitim, denetimli eğitim ve denetimsiz eğitim olarak ikiye ayrılır. Denetimli yöntemle ağa giriş çıkış yöntemi ile eğitim verilmektedir. İkinci adım, bu sefer bağlantı ağırlıklarının sabitlendiği ve buna karşılık gelen çıktının alındığı işlemdir.

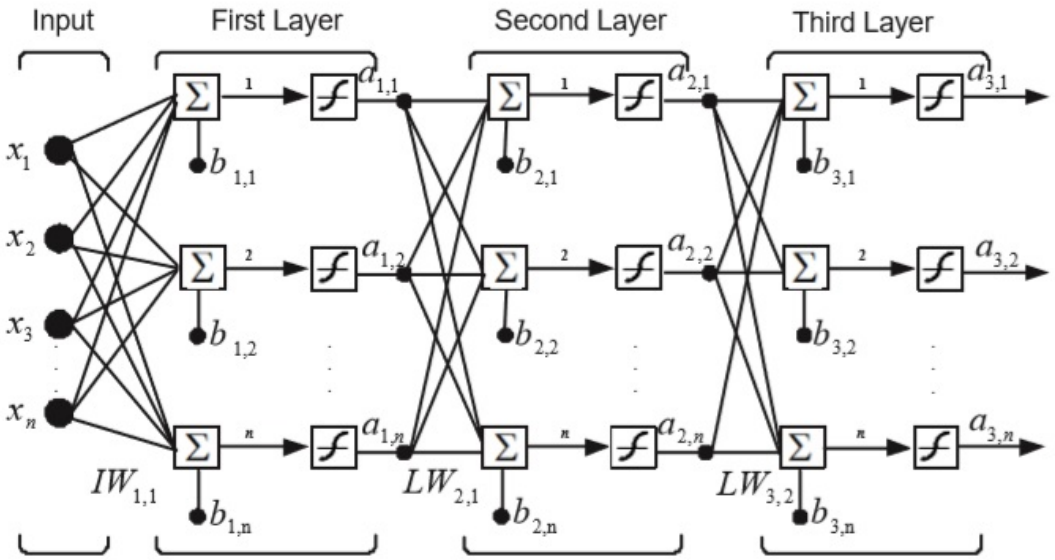
3.3.1. Çok Katmanlı Perceptron Ağı[1]

Sinir ağlarını modellemek ve tahmin etmek için en yaygın ve etkili sinir ağı tekniklerinden biri, birçok çalışmada temel model olarak kullanılan Çok Katmanlı Perceptron ağıdir (MLP) (Bui vd., 2016; Tien Bui vd., 2017). MLP, karmaşık ve doğrusal olmayan gerçek dünya olaylarını modellemek için çok yüksek bir yeteneğe sahiptir (Sadowski vd., 2018) (Pham ve diğerleri, 2016) (Bui vd., 2015). Genel olarak, ileri beslemeli MLP sinir ağı, Şekil 3.23'de gösterildiği gibi bir giriş katmanı, bir veya daha fazla gizli katman ve bir çıkış katmanı içerir (Kavzoglu ve Mather, 2003).

Genel olarak, girdi katmanı düğümlerinin sayısı, veri kaynağında seçilen faktörlerin sayısına bağlıdır. Gizli katmanlar hesaplama için kullanılır ve çıktı katmanı modelleme amacını temsil eder. Gizli katmanın her bir düğümü, giriş katmanının tüm düğümleriyle ve gizli katmandaki tüm düğümlerle bağlantılı olmalıdır. MLP'nin eğitim uygulaması iki adıma ayrılabilir: ileri ve Geri Yayılım algoritması kullanılarak geriye doğru yayılma (Bui vd., 2015).

İleri beslemeli bir MLP sinir ağının mimarisine bir örnek, Şekil 3.20'de gösterilmektedir, burada her transfer fonksiyonu "ai", önceki alt ağ topolojisinin katkısına bağlıdır.

[1] Multilayer perceptron



Şekil 3.20. İleri Beslemeli MLP yapısı (Azzini, A., ve Tettamanzi, A, 2006).

Şekil 3.20'de verilen örnek için tanımlanan transfer değerlerinin ifadeleri burada tanımlanmıştır.

$$a_1 = f_1 (IW_{1,1} * x + b_1) \quad (3.11)$$

$$a_2 = f_2 (LW_{2,1} * a_1 + b_2) \quad (3.12)$$

$$a_3 = f_3 (LW_{3,2} * a_2 + b_3) \quad (3.13)$$

Her gizli katmanda elde edilen transfer fonksiyonu, Denklem 3.14'te bildirilen ifade ile ardışık katman için girdi olacaktır.

$$a_3 = f_3(LW_{3,2} f_2(LW_{2,1} f_1(IW_{1,1} * x + b_1) + b_2) + b_3) \quad (3.14)$$

3.3.2. MLP İçinde İşleme Elemanlarının

MLP nöronları veya işleme elemanları (PE'ler), önceki katmandan girdi toplamak ve bir sonraki katmana çıktı sağlamak için gizli katmandaki PE'lerin davranışını belirlemek

için farklı transfer fonksiyonları kullanabilir. Axon, sadece girdi ve çıktı aktivitesi arasında bir kimlik haritası gerçekleştirir. Linear-Axon, eğim ve ofset kontrolüne sahip bir lineer Axon kullanır. Eğim, "β" ve uyarlanamayan bir parametre tarafından kontrol edilir. Sigmoid-Axon, katmandaki her nörona ölçeklenmiş ve önyargılı bir sigmoid fonksiyonu uygular. Katmandaki her nöron için değer aralığı 0 ile 1 arasındadır. Linear-Sigmoid-Axon, sigmoidin ara kısmını bir eğim çizgisi "β" ile değiştirir ve onu sigmoidin parçalı doğrusal bir yaklaşımı haline getirir. Tanh-Axon, katmandaki her nörona bir bias ve hiperbolik tanjant (tanh) fonksiyonu uygular. Bu, katmandaki her nöron için değer aralığı -1 ile 1 arasındadır. Linear-Tanh-Axon, tanh'ın ara kısmını bir eğim çizgisi "β" ile değiştirir, bu da onu tanh'ın parçalı doğrusal bir yaklaşımı haline getirir. Tablo 3.2, PE'ler için transfer fonksiyonları gösterir (URL-1, 2021).

Tablo 3.2. Farklı işleme elemanları transfer fonksiyonları (URL-1, 2021)

Processing elements	Transfer functions
Axon	$f(x_i, w_i) = x_i$
LinearAxon	$f(x_i, w_i) = \beta x_i + w_i$
SigmoidAxon	$f(x_i, w_i) = \frac{1}{1+\exp[-x_i^{\text{lin}}]}$ where $x_i^{\text{lin}} = \beta x_i$ is the scaled and offset activity
LinearSigmoidAxon	$f(x_i, w_i) = \begin{cases} 0 & x_i^{\text{lin}} < 0 \\ 1 & x_i^{\text{lin}} > 1 \\ x_i^{\text{lin}} & \text{else} \end{cases}$ where $x_i^{\text{lin}} = \beta x_i$ is the scaled and offset activity
TanhAxon	$f(x_i, w_i) = \tanh[x_i^{\text{lin}}]$ where $x_i^{\text{lin}} = \beta x_i$ is the scaled and offset activity
LinearTanhAxon	$f(x_i, w_i) = \begin{cases} -1 & x_i^{\text{lin}} < -1 \\ 1 & x_i^{\text{lin}} > 1 \\ x_i^{\text{lin}} & \text{else} \end{cases}$ where $x_i^{\text{lin}} = \beta x_i$ is the scaled and offset activity

3.3.3. Öğrenme Kuralları

MLP sinir ağı, Momentum, Quickprop, Conjugate Gradient ve Delta-Bar-Delta için farklı öğrenme kuralları sağlayabilir. Her öğrenme kuralı, her "epoch"dan sonra nöronların ağırlıklarını güncellemek için belirli bir fonksiyon uygular. Tablo 3.3, ağırlıkların güncellenmesi için farklı öğrenme kurallarının denklemlerini göstermektedir. Burada "ρ" ağırlık hareketinin mevcut yönüdür, "w" ağırlıklardır, "β", yeni konjugat yönünü oluşturmak için geçmiş yönün ne kadarının gradyanla karıştırıldığını belirleyen bir parametredir ve "g" gradyandır (geri yayılım bilgisi). a'nın denklemi, " ρ" yönü boyunca minimum Ortalama kare hatasını (MSE) bulmak için "α" satırı aramasıdır. "$\mathbf{J^p(w)}$", hata vektörünün Jacobian matrisidir, "$\mathbf{E^p(w)}$", "**ρ**" modeli için "w" içinde değerlendirilir ve "I", identity matrisidir. "μ" parametresi her adımda artırılır veya azaltılır (URL-1, 2021).

Tablo 3.3. Öğrenme kurallarının ağırlık güncelleme denklemleri (URL-1, 2021)

Learning rules	Weight update equations
Step	$\Delta w_i(n+1) = n_i \nabla w_i(n)$
Momentum	$\Delta w_i(n+1) = n_i \nabla w_i + \rho \Delta w_i(n)$
Conjugate gradient	$\Delta w = \alpha(n) p(n)$ $p(0) = -g(0)$ $p(n+1) = -g(n+1) + \beta(n) p(n)$ $\beta(n) = \frac{g^T(n+1) g(n)}{g^T(n) g(n)}$ $\alpha(n) = \arg\min(E(w(n) + \eta p(n))$
Quickprop	$\Delta^{(k)} w_{ij} = \Delta^{(k-1)} w_{ij} \left(\frac{\nabla_{ij} E^{(k)}}{\nabla_{ij} E^{(k-1)} - \nabla_{ij} E^{(k)}} \right)$
Delta-Bar-Delta	$\Delta w_i(n+1) = n_i \nabla w_i + \rho \Delta w_i(n)$

"Step", ayar için tahmin edilen yönde adımlar atarak ağın performans yüzeyinin altını bulmaya çalışır. Ağ öğrenimi, çok büyük seçilirse ıraksama veya salınım yapabilir ve adım boyutu küçükse çok yavaş olabilir."Momentum", gradyan inişini bir miktar ataletle (momentum parametresi, ρ) sağlar, böylece aşağı için ortalama tahmin olan bir yön boyunca hareket etme eğiliminde olur. En büyük fayda, yerel minimum lardan kurtulma yeteneğidir. "Delta-Bar-Delta", bir performans yüzeyi aramak için uyarlanabilir bir adım boyutu prosedürüdür. Momentum ve adım boyutu, PE'deki hatanın önceki değerlerine göre uyarlanır. Geçmiş ağırlık ve güncel güncellemelerin her ikisi de aynı işarete sahipse öğrenme oranı doğrusal olarak artar. Bunun nedeni, hatayı azaltmak için ağırlık aynı yönde hareket ettirilirse, daha büyük bir adım boyutuyla oraya daha hızlı ulaşacaktır. Güncellemeler farklı işaretlere sahipse, ıraksama önlemek için öğrenme oranı geometrik olarak azalır. Bu yöntem aşağıdaki adım boyutu güncelleme denklemine sahiptir:

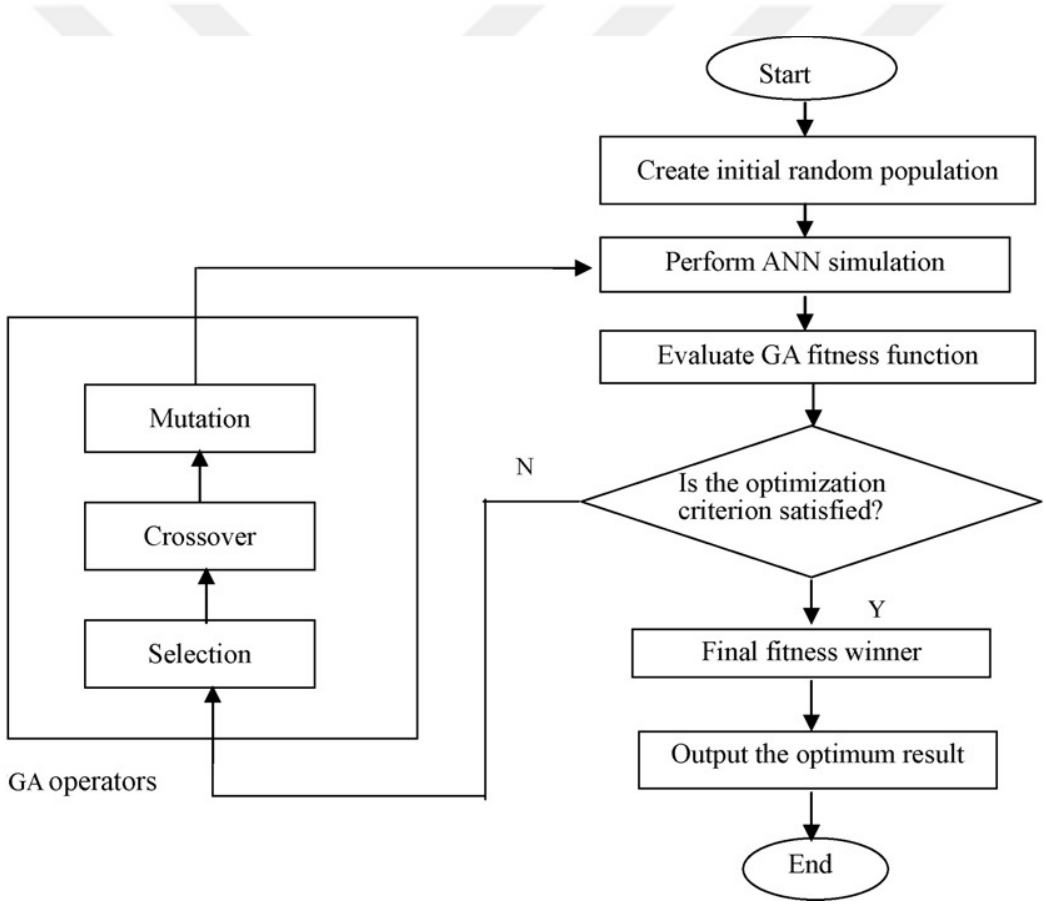
$$\Delta\eta_i(n) = \begin{cases} \kappa & S_i(n-1)\,\nabla w_i(n) > 0 \\ -\beta\eta_i(n) & S_i(n-1)\,\nabla w_i(n) < 0 \\ 0 & otherwise \end{cases} \tag{3.15}$$

Burada $S_i(n) = (1-\lambda)\ \nabla w_i(n-1) + \lambda S_i(n-1)$, κ=toplama sabiti, β= çarpımsal sabit, λ= yumuşatma faktörüdur. "Conjugate Gradient", ağırlık güncellemesini belirlemek için performans yüzeyinin ikinci türevlerini kullanan ikinci dereceden yaklaşıklık yöntemidir.

3.3.4. GA ile YSA Mimarisinin Optimizasyonu

Belirli bir uygulama için uygun bir sinir ağının seçimi esastır. En yaygın olarak kullanılan yapay ağlardan biri, çok çeşitli gerçek dünya problemlerini çözmek için kullanılan MLP sinir ağıdır. MLP'lerde, MLP ağının mimarisinin tasarımında katman sayısı ve işlem elemanlarının (nöronlar) sayısının belirlenmesi, dikkate alınması gereken önemli kararlardır. Aynı zamanda, katman sayısı ve işlem elemanlarının sayısını belirlemek için belirli kurallar yoktur ve bunlar deneme yanılma yoluyla belirlenir (Anderson ve McNeill, 1992), (Castillo vd., 2000). GA, optimal çözümleri bulmak için kullanılabilecek bir yöntemdir. GA algoritmaları, sinir ağlarının parametrelerini optimize etme yeteneklerini göstermiştir (Castillo vd., 2000). Birçok çalışmada GA, yapay sinir ağlarının parametrelerini optimize etmek için kullanılmıştır (Maaranen vd., 2003) (Correa

vd., 2011) (Ramezanian vd., 2019) (Lorencin vd., 2019) (Ilbeigi vd., 2020) (Sreedharan vd., 2020) (Bahiraei vd., 2020). Bu nedenle, bu çalışmada, minimum hesaplama hatası ile en uygun mimariyi elde etmek için bu parametreleri optimize etmek için GA kullanıldır. YSA mimarisinin GA algoritması ile optimizasyonunun ana hatları şekil 3.21'te ve bu işlemin sözde kodu şekil 3.22'te verilmiştir. İlk adımda, rastgele bir başlangıç popülasyonu oluşturulur, ardından karşılık gelen YSA oluşturulur ve YSA modeline dayalı uygunluk fonksiyonu, tüm ilk bireyler için uygunluğu (mean square error) hesaplamak için kullanılır. Daha sonra genetik operatörler (seçim, çaprazlama ve mutasyon) yeni nesli yeniden üretmek için kullanılır. İşlem maksimum nesil sayısına ulaşılana kadar tekrarlanır.



Şekil 3.21. YSA mimarisinin GA ile optimizasyonunun akış şeması (Shen, C., vd., 2007)

Initialize the population, either by generating new random individuals or by loading a previously saved population

Create the corresponding ANN

Evaluated initial population according to the fitness function (calculate its mean square error (MSE))

end for

Save the best individual as the best-so-far individual with the best (lowest) fitness value

While not termination condition **do**

Apply genetic operators to each network

(

Selection

Crossover

Mutation

)

Decode each new genotype into the corresponding ANN

Compute the fitness value for each ANN

Save statistics of the generation

end while

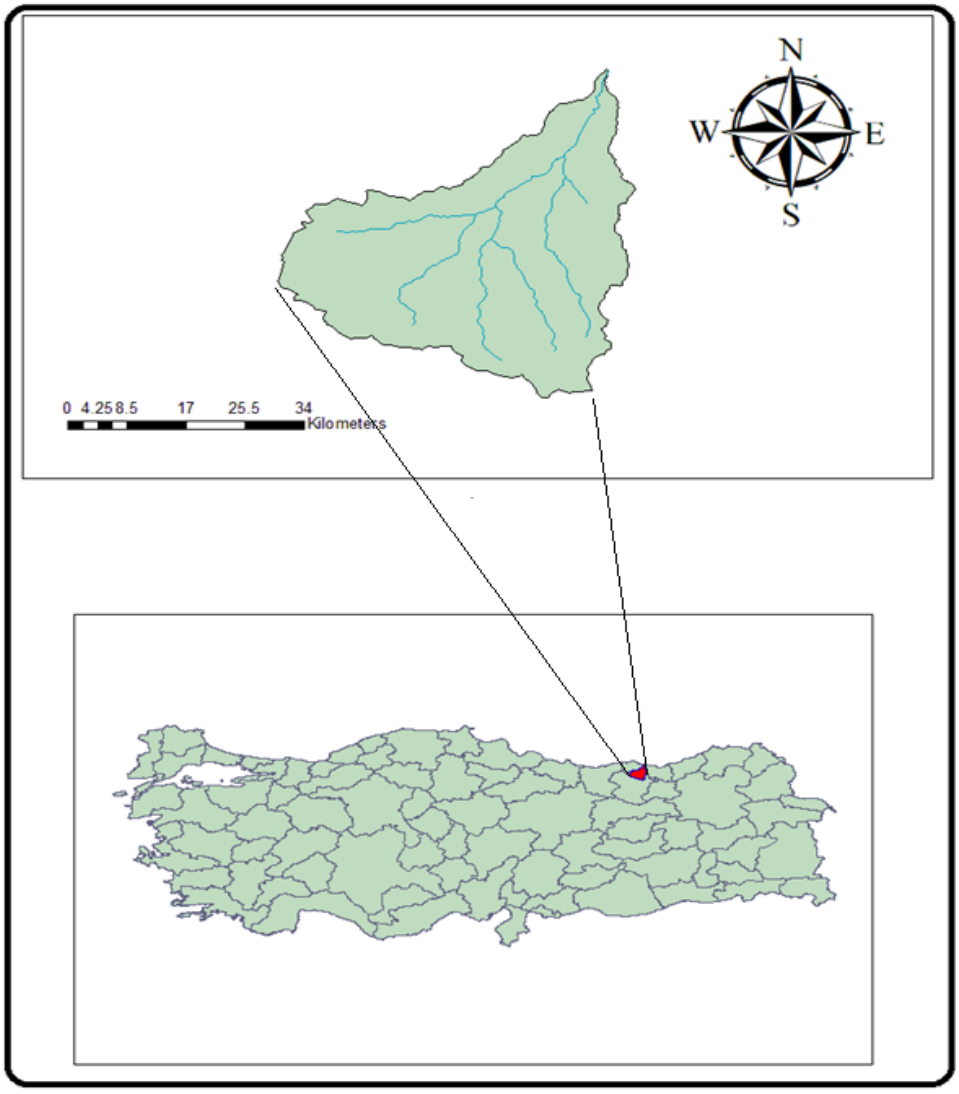
Return the best individual found during the evolution (The best architecture for ANN)

Şekil 3. 22. GA işlemi ile YSA mimarisinin Optimizasyonunun sözde kodu.

4. YAPILAN ÇALIŞMALAR VE BULGULAR

4.1. Çalışma Alanı

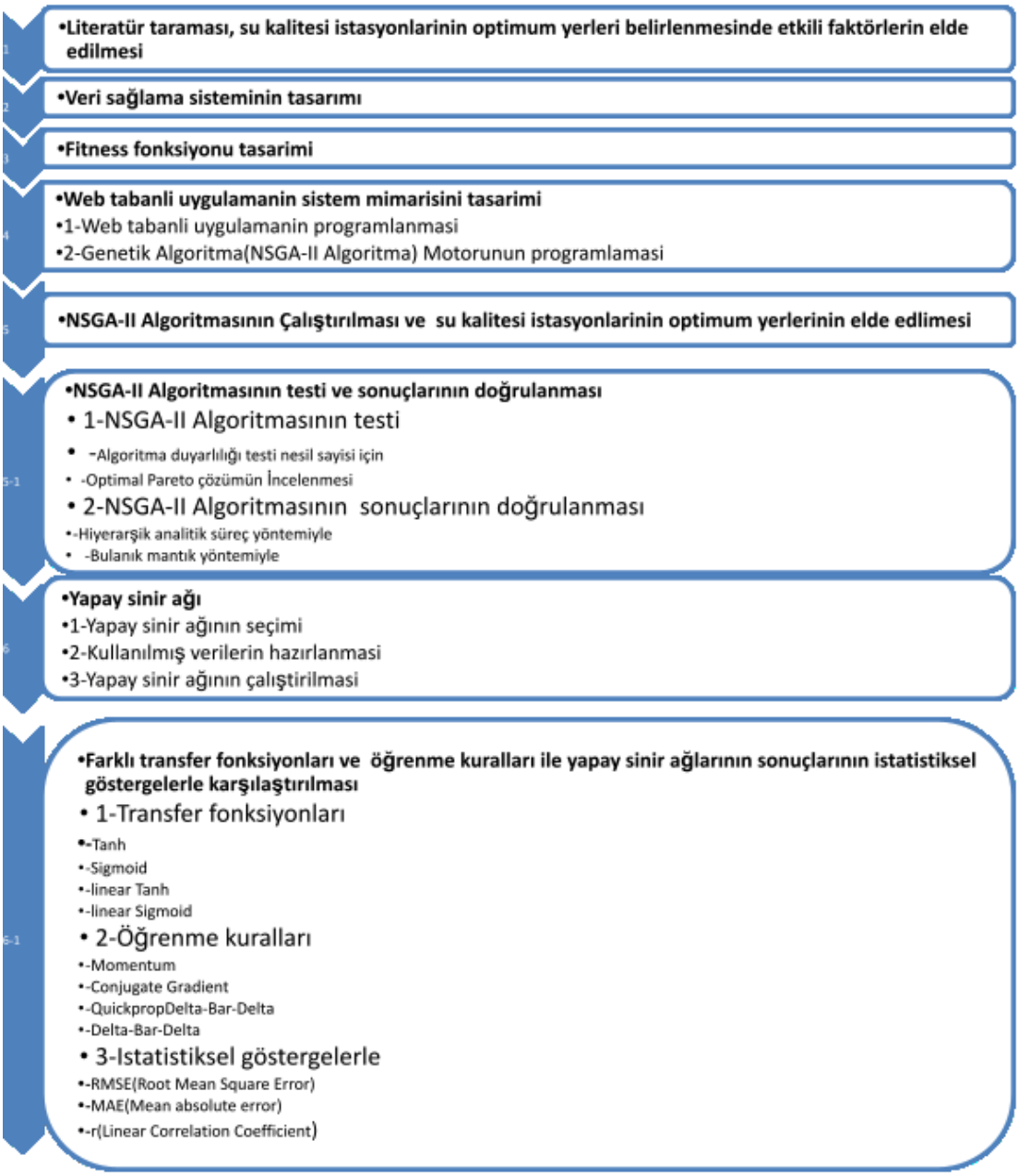
Bu tezde, Türkiye'de Trabzon ilindeki Değirmendere havzasını örnek olay incelemesi olarak seçtik. Değirmendere havzası 1061 kilometrekarelik bir alana sahiptir. Bu havza 41 ° 00 'Doğu ve 39 ° 45' Kuzeyde yer almaktadır.



Şekil 4.1. Çalışma alanı

4.2. Araştırma Adımları

Araştırmanın gerçekleştirilmesi için gerekli adımlar genellikle Şekil 4.2'da gösterilmektedir



Şekil 4.2. Araştırma yapmak için adımlar

4.3. Kullanılan Veriler

Bu çalışmada aşağıdaki veriler kullanılmıştır:

1-1/25000 topografik sayısal haritalarlari

Köy hizmetleri Genel İdaresi veriler, yönetiminden alındı

2- Meteorolojik veriler

-Ortalama aylık sıcaklık

T.C.TARIM VE ORMAN BAKANLIĞI Meteoroloji Genel Müdürlüğü'nden alındi (https://mgm.gov.tr/veridegerlendirme/il-ve-ilceler-istatistik.aspx?k=undefined&m=TRABZON).

3- Bina ile ilgili Kullanıcı verileri(Konut, ticari, fabrika,yollar, madencilik vs)

Bu veriler Trabzon Belediyesinden alındı.

4- Arazi kullanım verileri(tarım alanları, hasarlı ormanlar ve vs)

T.C.TARIM VE ORMAN BAKANLIĞI'nden alındı.

(http://geodata.ormansu.gov.tr).

4.4. Nehir Suyu Kalitesi İzleme Hedeflerinin Belirlenmesi

Nehir kalitesi sensör ağının etkin tasarımı, çalışmanın hedeflerine ve nehir suyunun nicel ve nitel koşullarına göre yapılır. İzleme hedefleri, operasyonel ve yönetsel ihtiyaçlara göre belirlenir ve su kalitesi standartlarının oluşturulmasına yardımcı olmayı, su kalitesi durumunu ve eğilimlerini belirlemeyi, kirli suların belirlemeyi, su kalitesi sorunlarının nedenlerini ve kaynaklarını belirlemeyi ve su kalitesi programlarını değerlendirmeyi içerebilir (US EPA, 2003). Bu çalışmada, hem nehir suyu izleme ağlarının geleneksel

hedefleri, nehir suyu kaynaklarının yönetimi için getirilen yeni hedefler hem de su kalitesi sensörlerinin kurulumu için uygun yerler dikkate alınmıştır.

4.4.1. Su Kalitesini İzlemenin Geleneksel ve Yeni Hedefleri

Su kalitesi izleme ağlarını izlemenin geleneksel hedefleri şunları içerir: 1- Su kalitesi parametrelerindeki zamansal değişimlerin uzun ve / veya kısa vadeli eğilimlerini anlamak 2- Her havza için belirtilen su kalite standartlarının ihlallerini izlemek için 3- Su kalitesi değişikliklerini etkileyen dış nedenleri ve kaynakları tanımlamak 4- Su kaynaklarının kullanımını desteklemek 5- Belirli bir süre boyunca hedeflenen bir soruşturma vasıtasıyla su kalitesindeki kısa vadeli değişimleri incelemek(Lettenmaier, 1979).

Ayrıca, su kaynakları yönetimi için TMDL ve bilgi sistemleri gibi son politika ve teknolojiler yeni yönetim gereksinimleri oluşturulmuş ve aşağıdaki gibi su kalitesi kontrol ağlarını izlemek için başka hedefler oluşturmuştur:

6- TMDL analizlerini gerçekleştirmek için her bir havza biriminden gelen kirlilik yüklerini tahmin etmek 7- TMDL'yi desteklemek için su kalitesi modellemesini kullanmak ve bilimsel su kalitesi yönetim fonksiyonları 8- Su kaynakları yönetimi için bilgi sistemleri kurmak (Parket vd., 2006).

Yukarıdaki hedeflere ek olarak, onarım ve bakım imkanı ve kayıtlı bilgilerine erişim ve doğru ölçüm ve kolay erişim ile birlikte maksimum verimliliğe sahip uygun yerlere su kalitesi sensörlerinin kurulması gerekmektedir.

4.4.2. Nehir Suyu Kalite Sensörlerinin Montajı İçin Uygun Yerlerin Özellikleri

Nehir suyu kalite sensörlerinin kuruluş yerini etkileyen en önemli faktörleri aşağıdaki gibidir:

4.4.2.1. Çevresel Faktörler

Çevresel faktörler, fiziksel, biyolojik, hidrodinamik ve antropolojik faktörleri içerir. Su kalitesi sensörlerinin kurulumunu etkileyen en önemli çevresel faktörlerden biri buz oluşumunun derecesidir. Genel olarak, donmaya maruz kalan alanların geçmişini bilmek önemlidir, çünkü su donması muhtemelen su kalitesi sensörlerinin çalışmayı durdurmasına neden olacaktır. Genel olarak, çalışma alanındaki buz oluşumunun geçmişini bilmek, istasyon tasarım amaçları, istasyon konumu, istasyon izleme ve istasyon bakım faaliyetleri için yararlıdır. En önemli çevresel-insan faktörlerinden biri, yerel olarak su kalitesini etkileyebilecek belediye veya endüstriyel atık su gibi özel insan faaliyetleridir. İnsan faaliyetlerinin etkisi iki şekilde olabilir: 1- Noktasal kaynaklar(sanayi alanlarından kanalizasyon ve hastanelerden gelen atık su gibi) 2- Noktasal olmayan kaynaklar(tarım arazisi , hasarlı orman alanları,maden alanları ve kentsel alanlar gibi) (EJ Miles, 2009).

4.4.2.2. Finansman - Bütçe Hususları

Maliyet, bir su kalitesi izleme programının tasarlanmasında kilit bir faktördür (Cavanagh vd., 1998). Bu durumda üç ana faktör dikkate alınmalıdır: 1. İnşaat maliyetleri 2. Bakım maliyetleri 3. Erişim maliyeti. İnşaat maliyetleri, şantiyeye bağlı olarak değişebilir. İzleme sistemi için planlanmış bakım faaliyetleri, muhtemelen su kalitesi izleme sensörlerinin temizlenmesini ve kalibre edilmesini içerecektir. Bakım sıklığı, sensörlerin çalışma oranına ve oranına göre sensör tipine, hidrolik ortama, mevsime, sensörlerin gücü için kullanılan enerji türüne (pil veya güneş enerjisi gibi) ve veri depolama kapasitesine göre değişiklik gösterir. Ve istasyona erişim maliyeti, istasyona erişmek için gereken aracın türüne, gereken personele, konuma olan mesafeye vb. Bağlıdır(EJ Miles, 2009).

4.4.2.3. Erişilebilirlik ve Güvenlik Sorunları

Erişim ve güvenlik sorunları, site seçiminde önemli bir rol oynayan sitenin iki önemli özelliğidir. İzleme istasyonları, izleme süresi boyunca mevcut olmalıdır. İstasyona erişim 1- kurallar, 2- yerin topografyası, 3- arazi sahibinin izni 4- hava koşullarından etkilenir. İstasyon her zaman erişilebilir olmalıdır, bu nedenle İstasyona erişim için hava ve su akış koşullarının olası etkilerini göz önünde bulundurmak önemlidir. Özel hava koşulları dikkate alınmalı, buz oluşumu gibi (istasyonlara ve güvenlik öğelerine erişim için). Kış koşulları çok sert ise, ekipmanı veya hatta istasyon platformunu çıkarmak gerekebilir. Personelin ve ekipmanın güvenliği en önemli önceliktir, bu nedenle minimum güvenlik gereksinimlerini karşılayan izleme alanlarının seçimine özel dikkat gösterilmelidir(EJ Miles, 2009).

Yukarıda bahsedilen izleme hedefleri nehir suyunun genel izlenmesi içindir ve her bir ağın özel hedeflerine göre tamamlanabilir ve diğer hedeflerle değiştirilebilir(EJ Miles, 2009).

4.5. Sorunu Çözmek İçin Fitness Fonksiyonu

Bu bölümde, bir önceki bölümde bahsedilen amaçlara göre, bu hedeflerden birkaçı seçilmiş ve bunlar için aşağıda belirtilen fitness fonksiyonu belirlendi.

4.5.1. Su Kullanımının Denetimi

Bu fonksiyon, hedef 4'e ulaşmak için tasarlanmıştır(Kentsel ve endüstriyel kullanım için su tüketimi gibi su tüketimi kaynaklarını izlemek için). Su tüketim kaynakları ile su kalitesi ölçüm istasyonları arasındaki mesafe ne kadar azsa o kadar iyidir. Diğer bir deyişle, bu mesafeyi azaltarak, su kalitesi sensörlerinin konumu için o konumu seçme olasılığı artar. Aşağıdaki fonksiyone gore:

f_1= *Min(Nehir ve su tüketimi kaynakları arasındaki öklid mesafesi)*

4.5.2. Kirlilik Kaynaklarının Gözlemlenmesi

Bu fonksiyon, hedef 3'ya ulaşmak için kirlilik kaynaklarını izlemektir. F2 fonksiyon, nehirdeki kirletici konsantrasyonunun tersine azaldığı varsayımına dayanır, bu nedenle sensörler kirlenme alanına ne kadar yakınsa, seçim şansı o kadar yüksek olur.

f_2 *=Min(Nehir ve suyu kirleten kaynaklar arasındaki Öklid mesafesi)*

4.5.3. Su Kalitesindeki Değişikliklerin İncelenmesi / Kirlilik Yüklerinin Tahmini

Bu fonksiyon, hedef 5 ve 6'ye ulaşmak için tasarlanmıştır.Bu fonksiyonda, her bir havzanın çıkışındaki su kalitesindeki değişikliklerin daha iyi gösterileceği varsayılır, bu nedenle su kalitesi sensörlerinin havza çıkışına olan mesafesi ne kadar yakınsa, seçilme şansları daha yüksektir.

f_3*= Min(Nehir ile her alt havzanın çıkışı arasındaki öklid mesafesi)*

4.5.4. Veri Transferi

Bu fonksiyon, sensörler tarafından kaydedilen verileri iletmek için sensörler tarafından kaydedilen verilerin daha hızlı aktarılması için tasarlanmıştır.Seçilen konumun cep telefonu servislerine (antenler) veya sabit telefona yakın olması gerekir. Bu nedenle, su kalitesi sensörlerinin bu hizmetlere olan uzaklığı ne kadar yakınsa seçim şansı da o kadar yüksek olur.

f_4*=Min (Nehir ile sabit telefon hatları veya cep telefonu antenleri arasındaki öklid mesafesi)*

4.5.5. Kolay Erişim

Bu fonksiyon, istasyonlara kolay ve güvenli erişimin ayrıca istasyonlerin kolay bakımı amacını karşılamak için tasarlanmıştır ve seçilen alanlarin mevcut yollara yakınlığını gösterir, istasyonlar yollara ne kadar yakınsa, o kadar yüksek seçilme şansı vardir.

f_5 =*Min(Nehir ve yollar arasındaki Öklid mesafesi)*

4.5.6. Fitness Fonksiyonlarını Birleştirmek

Evrimsel çok amaçlı optimizasyon (EMO) yöntemleri, optimizasyon için iki veya üç hedef seçerken en iyi performanslarını gösterirler (Deb, 2011). MOO'da çatışan birkaç hedef vardır ve bu nedenle hepsi aynı anda optimumlarına ulaşamazlar. Bu nedenle, problemin çözümleri, belirli bir problem için en iyi çözümün seçilebileceği bir dizi değiş tokuş çözümü (bir Pareto-optimal cephe veya Pareto-optimal çözümler) oluşturur. Pareto-optimal cepheler genellikle bu amaç için görselleştirilir, çünkü bu şekilde çözümler arasında Pareto-optimal cephedeki konumlarına göre bir karşılaştırma biraz daha kolay hale gelir. Yalnızca iki hedef (veya amaç fonksiyonu) olduğunda bir Pareto-optimal cepheyi görselleştirmek kolaydır, ancak bir cepheyi ikiden fazla amaç fonksiyonu için görselleştirmek zor bir görev haline gelir (Madetoja vd., 2008). . Ayrıca, büyük boyutlu bir Pareto-optimal cepheyi temsil etmek için üstel olarak büyük bir popülasyon büyüklüğüne ihtiyaç vardır. Bu, bir EMO prosedürünü yavaşlatır ve hesaplama açısından daha az çekici hale getirir (Deb, 2011). Pareto kavramı ve baskın olmayan çözümlerle çalışan EMO yöntemlerinde hedef sayısının arttırılmasıyla ilgili bir diğer sorun, rastgele oluşturulmuş bir popülasyonun üyelerinin çoğunun birbirine baskın olmama eğiliminde olmasıdır (Li B vd., 2015) (Ishibuchi vd., 2008). Bu bir ölçeklenebilirlik (scalability) sorunu yaratır. Ölçeklenebilirlik sorunu, daha iyi çözümler seçmek için baskın olmayan ölçekle çözümlerin birbirine göre karşılaştırılabilirliğini azaltır, bu da bir EMO algoritmasının performansında durgunluğa neden olur.

EMO araştırmacıları, çok amaçlı problemlerle uğraşırken iki farklı yaklaşım benimsemiştir: 1- Kısmi bir küme bulma 2- Gereksiz hedefleri belirleme ve ortadan kaldırma. İlk yöntemde, çok sayıda amacı olan bir problemde tam Pareto-optimal cepheyi bulmak yerine, EMO prosedürleri Pareto-optimal cephenin sadece bir kısmını bulmak için

kullanılabilir. Bu, tercih bilgilerinin çeşitli yollarla gösterilmesiyle sağlanabilir. Referans noktası tabanlı EMO (reference point-based EMO), ışık huzmesi arama(light beam search), önyargılı paylaşım yaklaşımları(biased sharing approaches) vb, bu amaçla önerilmiştir. İkinci yöntemde ise birbiriyle çelişmeyen amaçların belirlenmesi ve bu amaçların ortadan kaldırılması veya birleştirilmesiyle amaç sayısının azaltılması yapılır (Deb, 2011). Bu araştırmada ikinci yöntemi seçtik ve bir önceki bölümde bahsedilen fonksiyonlari iki grupta birleştirdik:

1. Nehir suyu kalitesini etkileyen faktörleri belirleme süresini en aza indirmek

Nehir suyu kalitesini etkileyen faktörleri hızlı bir şekilde belirlemek için birinci, ikinci ve üçüncü fonksiyonların birleştirilmesiyle elde edilir ve aşağıdaki fonksiyonla hesaplanir

F1(Birincil hedef fonksiyonu)=Min(Nehir ve kirleten kaynaklar,su tüketimi kaynakları,her alt havzanın çıkışı arasındaki öklid mesafesi)

2. Maliyeti en aza indirmek

Dördüncü ve beşinci fonksiyonların birleştirilmesiyle elde edilir ve aşağıdaki fonksiyonla hesaplanır

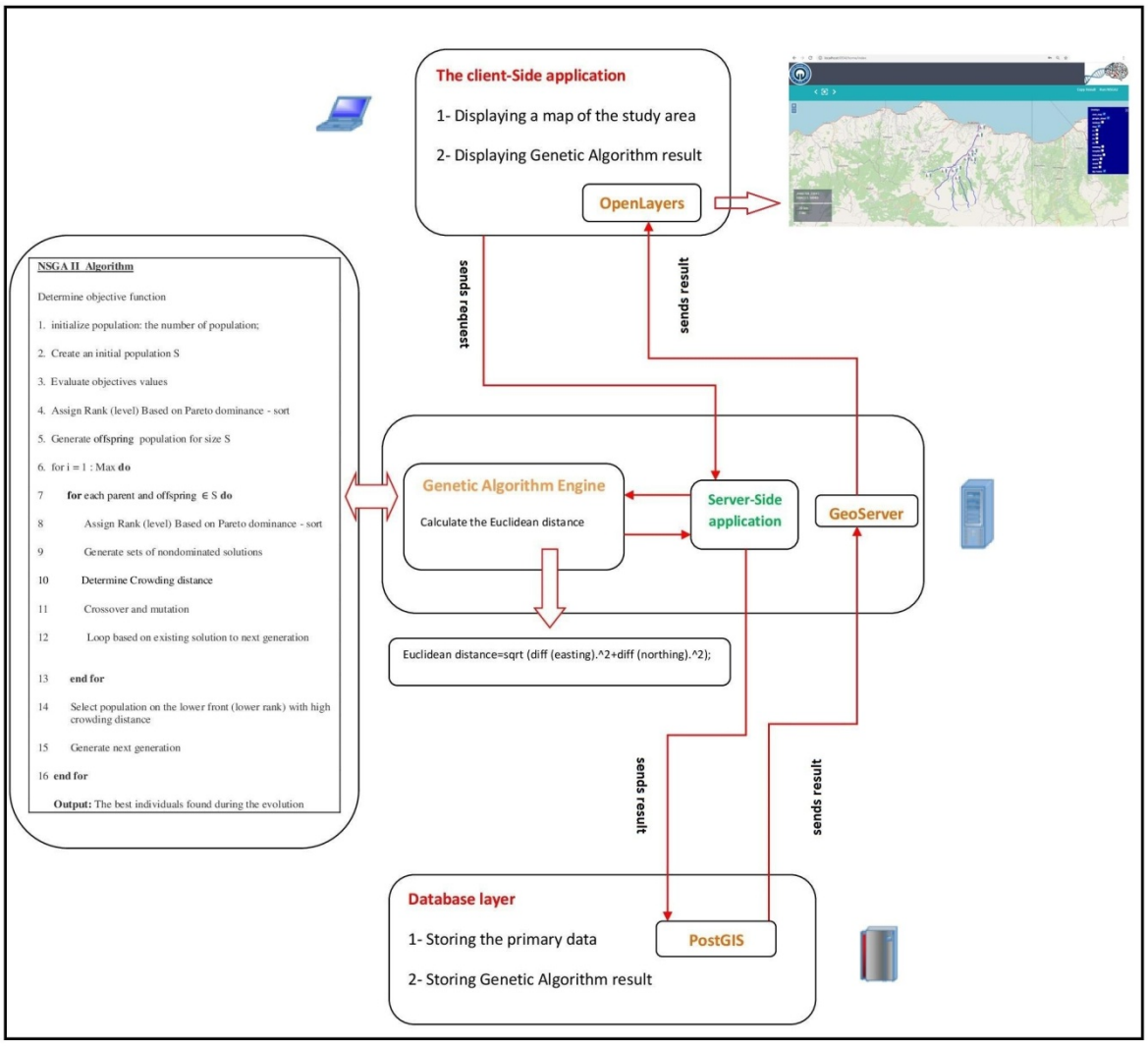
F2(İkinci hedef fonksiyonu)=Min(Nehir ve yollar ,telefon hatları veya cep telefonu antenleri arasındaki Öklid mesafesi)

4.6. Web Tabanlı Uygulamanın Sistem Açıklaması

Bu çalışmada mekansal verilerin depolanması için açık kaynak veritabanı (PostgreSQL) kullanılmış ve haritaların web üzerinde gösterilmesi için açık kaynak yazılım (GeoServer) ve programlama için Visual Studio bedava versiyonu (Community) programlama ortamı belirlenmiştir.

4.6.1. Web Tabanlı Uygulamanın Sistem Mimarisi

Şekil 4.3'e göre sistem mimarimiz üç farklı katmandan oluşmaktadır. Tasarlanan web tabanlı uygulaması, GA'ları yürütme ve tasarım sonuçlarını görselleştirme becerisine sahiptir. İlk katmanda, web tabanlı uygulaması İstemci Tarafı uygulamasını içerir. Bu katmanda, çalışma alanının haritasını olağan webGis yetenekleriyle görüntülemekten sorumlu olan Openlayers kitaplığı kullanılarak bir harita yerleştirilir (yakınlaştırma, uzaklaştırma, ...) ve ayrıca İşleme sonuçlarını haritada gösteriyor.İşlemeyi başlatmak için kullanıcı, isteğini İstemci tarafı uygulaması aracılığıyla sunucuya gönderir. İkinci katman Sunucu Tarafı uygulamasını içerir ve C # dili ile geliştirilmiştir. Bu katman, birinci katmandan bir talep aldığında, ihtiyaç duyduğu tüm verileri veritabanı olan üçüncü katmandan (nehirlerin, yolların, kirlilik kaynaklarının mekansal verileri vb.) Çağırarak bize verir. bağımsız bir modül olarak geliştirilen ve ikinci katmana dahil edilen web uygulama motoru (NSGA-II). Bu motor girdileri yakalar ve işledikten sonra sonuçları döndürür ve bunları üçüncü katman veritabanı katmanında saklar, ayrıca Geoserver bu katmanında mekansal verileri görüntülemek ve görselleştirmek için kullanıldı, Kullanıcı arayüzüne gömülü harita talebini Geoserver'a gönderdikten sonra Geoserver, Veritabanından mekansal verileri (temel harita, nehirler vb. gibi çalışma alanı) alır ve ardından oluşturulan haritayı ilk katmana gönderir(PNG gibi bir grafik dosyası biçiminde). Üçüncü katman veya veritabanı katmanı, birincil verilerin, ikinci katmandan üretilen sonuçların depolanmasından ve ayrıca birinci ve ikinci katmanların gerektirdiği mekansal verilerin gönderilmesinden sorumludur. Bu katmanda, veritabanı yönetiminde Açık kaynak veritabanı, postgre SQL kullanılmıştır.



Şekil 4.3. Sistem Mimarisi

4.6.2. GA Motoru

Bu çalışmada tasarlanan GA motorunun, tasarlanan web uygulamasının diğer bölümleriyle bütünlüğünü korumak için Visual studio.net programlama ortamında C # programlama dili kullanılmıştır. Ve kullanıcının rahatlığı için, GA motoru için veri girişi, uzamsal veriler için kullanılan standart ve yaygın format olan bir "shapefile" formatıdır İlk aşamada, bu araştırmada kullanılan GA motorunun bir ilk popülasyonun oluşturması gerekiyor ki bu GA motorunun verimliliğini artırmak için yapılan tasarımda, başlangıç popülasyonunu iki şekilde tanımlamak mümkündür. İlk durumda, ilk popülasyonun üretimi

uzmanların görüşü ile yapılır ve UTM noktalarının koordinatlarından GA'nın motoruna bir metin dosyası şeklinde tanıtılır Ve ikinci durumda, ilk popülasyonun üretimi rastgele yapılır. Mesafe hesaplaması ile ilgili olarak F1 ve F2 kriterleri için Öklid minimum mesafesi dikkate alınmıştır. Ayrıca sonuçlar burada, istemci tarafında(client-side) kullanım kolaylığı için bir coğrafi koordinat sistemine dönüştürülür.

4.6.3. Entegrasyonun Yönleri

GA'lar optimizasyon için güçlü araçlardır. Birçok çalışma GA'ları kullanmış ve GA'ların optimal yer seçimi için yeteneklerini göstermişler. Ancak, bu çalışmalar Çok Amaçlı GA'yı (MOGA) CBS ile entegre etmeyi amaçlamamışlar. Buradaki entegrasyon, MOGA tarafından optimal yerleri bulma ve sonuçları web'deki haritada gerçek zamanlı olarak görselleştirme sürecini, açık kaynak araçları tarafından geliştirilen bağımsız bir web tabanlı uygulamada ve mekansal verilerle çalışma becerisini entegre etmek anlamına gelir. Önceki çalışmalarda bu entegrasyon, bir MOGA'yı tek amaçlı bir GA'ya dönüştürerek, masaüstü tabanlı yazılımda (Li ve Yeh, 2005) veya MATLAB gibi ticari yazılımlar kullanılarak (Handayanto vd., 2015) yapılmıştı. CBS mekansal analiz için güçlü bir araçtır. CBS yeteneklerini GA'larla birleştirmek, kullanıcıların karmaşık hesaplamalar yaparken mekansal kararlar için CBS özelliklerini kullanmalarına olanak tanır. Başlangıçta, CBS masaüstü tabanlı uygulamalar kullanıyordu. Ancak, kullanıcıların önce programı yüklemeleri gerektiğinden kullanımı zordu. Bugün, İnternet'in hızlı büyümesi ve bilgisayarlar, tabletler ve akıllı telefonlar gibi çeşitli cihazlardan Web'e erişim ile birlikte birçok uygulama geliştiricisi, web'den erişilebilen hizmetleri kullanıyorlar. Web hizmetleri, web uygulamalarının entegrasyonunu ve birlikte çalışabilirliğini ve makineler arasındaki etkileşimi sağlar. Hizmet odaklı mimari (SOA), artık CBS uygulamalarının geliştirilmesi için ana modeldir. Geliştiriciler ve araştırmacılar, web hizmeti teknolojisini kullanarak web üzerinde mekansal uygulamalar geliştirdiler. Bu teknoloji kullanılarak uygulamanın tüm detayları kullanıcıdan gizlenir ve kullanıcının sadece bir web tarayıcısına ihtiyacı olur (Delipetrev, vd., 2014). WebGIS çok ilgi çekicidir, çünkü internet ile belirli bir yeri yayınlayarak herkes için kolayca erişilebilir hale getirir ve şu anda herhangi bir CBS geliştiricisi için ilk tercihtir. Bu tip CBS üzerinde birçok araştırmacı çalışmıştır (Cömert ve Akinci, 2003), (Cömert, 2004), (Cömert ve Akinci, 2007), (Cömert vd., 2008),

(Fotheringham vd., 2005), (Urbieta, 2014), (Fustes, 2014), (Fotheringham vd., 2005). Bu çalışmada, GA + GBS özelliklerini, MOGA ile en uygun yerleri seçme ve sonuçları bir web haritası üzerinde açık kaynak araçlarla otomatik olarak görüntüleme işlemini gerçekleştiren bir web servisine entegre ettik.

4.7. Web Uygulamasını Programlama ve Çalıştırması

Bu çalışmada mekansal verilerin depolanması için açık kaynak veritabanı (PostgreSQL) , ve haritaların web üzerinde gösterilmesi için açık kaynak yazılım (GeoServer) ve programlama için Visual Studio bedava versiyonu(Community) programlama ortamı , ve Microsoft'un web sayfası üretimi için son modeli olan "Asp.net MVC'yi" kullanılmış. Ayrıca gelişmiş yetenekleri programlamak için C # programlama dili kullanıldı. Sistem için seçilen mimari nedeniyle, bazı coğrafi Bilgi Sistemiyetenekleri istemci tarafında(client-side) programlanmalıdır, bu amaçla JavaScript programlama dili, istemci tarafında(client-side) kullanıldı. CBS Yetenekleri bu programlama dili ile hayata geçirilebilmesi için bu amaçla geliştirilmiş kütüphanelerden yararlanılabilir. Bu amaç için JavaScript kullanılarak uygulanan en önemli kitaplıklardan ikisi Openlayers ve Leaflet'tir. Bu araştırmada, istemci tarafında CBS yeteneklerini programlamak için açık kaynak kütüphanesi Openlaers kullanıldır.

4.7.1. Kullanılmış Verilerin Ön İşlenmesi

Bu bölümde, toplanan tüm mekansal verilerin koordinat sistemi ilk olarak UTM koordinat sistemine değiştirildi. Bir sonraki adımda, ortalama sıcaklığı sıfır derecenin altında olan alanları çalışma alanından çıkarmak için çalışma alanının 1927'den 2020'ye kadar olan uzun vadeli ortalama aylık sıcaklık verileri incelendi (Tablo 4.1). Çalışma alanındaki ortalama uzun vadeli aylık sıcaklık incelendiğinde, çalışma alanında sıfır derecenin altında ortalama sıcaklık gözlenmemiştir. Bu nedenle sıcaklık açısından hiçbir bölge çıkarılmadı.

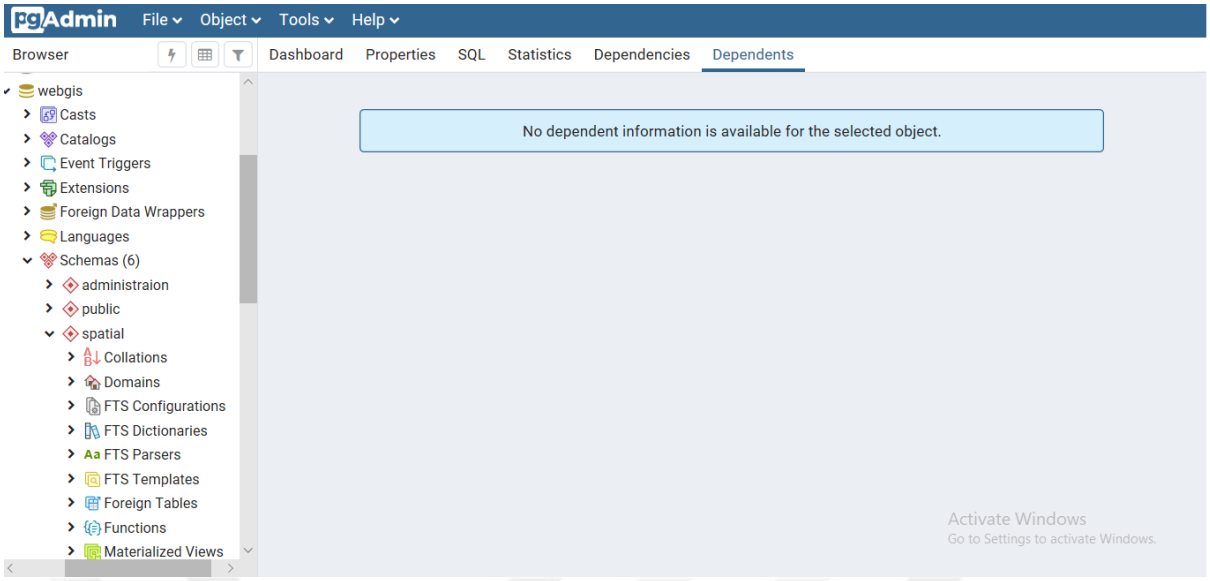
Tablo 4.1. Ortalama aylık sıcaklık (Ölçüm Periyodu 1927 - 2020)

Ay	Ocak	Şubat	Mart	Nisan	Mayıs	Haziran	Temmuz	Ağustos	Eylül	Ekim	Kasım	Aralık	Yıllık
Ortalama Sıcaklık (°C)	**7.4**	**7.3**	**8.4**	**11.7**	**15.9**	**20.2**	**23**	**23.4**	**20.4**	**16.7**	**13**	**9.6**	**14.7**
Ortalama En Yüksek Sıcaklık (°C)	**10.8**	**10.8**	**12**	**15.5**	**19.2**	**23.2**	**26**	**26.6**	**23.8**	**20.1**	**16.5**	**13**	**18.1**
Ortalama En Düşük Sıcaklık (°C)	**4.7**	**4.4**	**5.4**	**8.7**	**12.9**	**17**	**19.9**	**20.4**	**17.4**	**13.7**	**10**	**6.7**	**11.8**

Bir sonraki adımda, çalışma alanının eğimi incelenmiş ve nehrin ana kollarının eğimi 35 dereceden az olduğu için hiçbir alan kaldırılmadir.

4.7.2. Verileri , PostgreSQL / PostGIS Coğrafi Veritabanına Aktarılması

Bu bölümde veriler, açık kaynaklı PostgreSQL / PostGIS Coğrafi Veritabanına aktarıldı. PostGIS, PostgreSQL nesne-ilişkisel veri tabanında coğrafi detayların kullanımını sağlar. PostGIS ile birlikte PostgreSQL sunucusu, ESRI SDE veya Oracle Spatial eklentisi gibi coğrafi bilgi sistemleri için bir mekansal sunucu olarak görev yapabilmektedir. PostGIS ayrıca, OGC'nin "Simple Features Specification for SQL" standardını da desteklemektedir (PostGIS, 2015). PostGIS, çok sayıda açık kaynaklı harita sunucusu tarafından kullanılmaktadır ve geniş çapta desteğe sahiptir (Ballatore et al,2011).

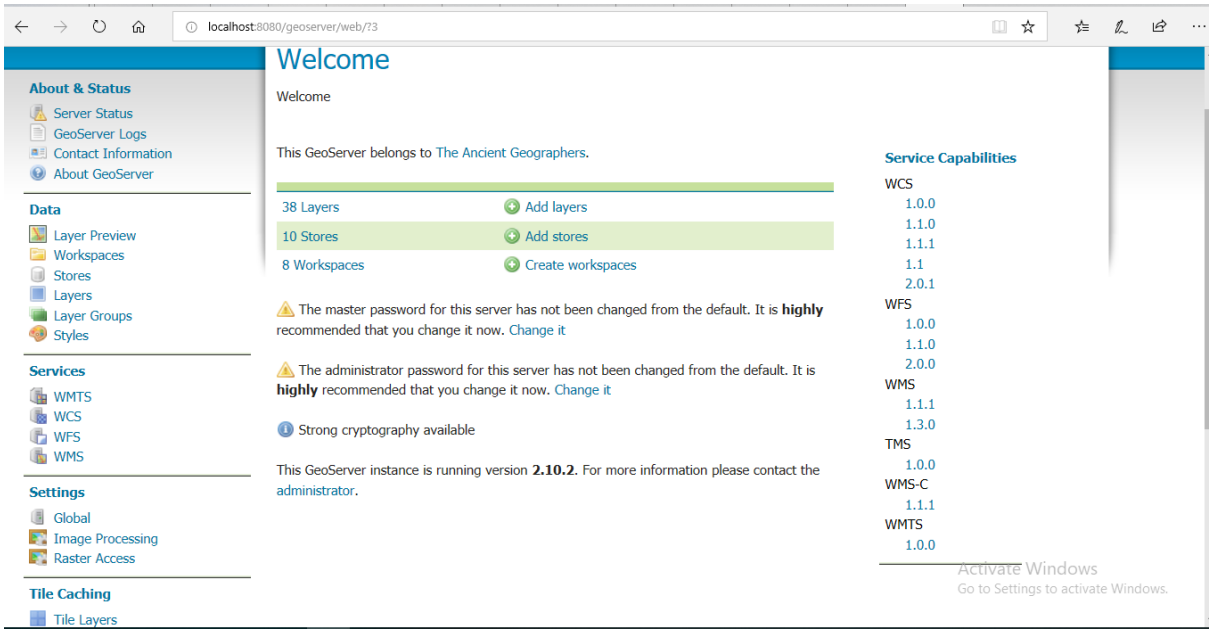


Şekil 4.4. PostgreSQL yazılım görüntü ortamı

4.7.3. GeoServer'ı PostgreSQL Veritabanına Bağlatılması

Bu bölümde, GeoServer, PostgreSQL veritabanına bağlandı ve veritabanına girilen veriler GeoServer'a tanıtıldı. GeoServer, kullanıcıların mekansal verileri görüntülemesine ve düzenlemesine imkan veren, Java temelli ve açık kaynaklı bir Web harita sunucusu yazılımıdır. GeoServer aynı zamanda, WFS, WMS ve WCS gibi, OGC tarafından geliştirilen açık standartları da desteklemektedir (GeoServer,2015) GeoServer, dünyanın farklı yerlerindeki kuruluşlardan oluşan özel bir grup tarafından geliştirilmekte ve geniş bir kitle tarafından desteklenmektedir. Bu şekilde yazılımın sürekli güncel kalması ve ihtiyaçlar doğrultusunda özelleştirilmesi kolay ve hızlı bir şekilde gerçekleşmektedir. Birgören (Birgören, M., 2009) yapılan web isteklerine sunucudan gelen yanıt sürelerini karşılaştırmış ve bu karşılaştırma sonucunda GeoServer yazılımının genel olarak piyasada var olan diğer rakiplerinden (Mapserver, ArcGİS Server) daha iyi yanıt sürelerine sahip olduğunu ortaya koymuştur. Ayrıca, GeoServer, haritaların görsel tasarımına yönelik diğer bir OGC standardı olan SLD/SE ve istemci tarafında detay servislerinin düzenlenmesini sağlayan WFS-Transactional standartlarını da desteklemektedir. GeoServer tarafından desteklenen raster ve vektör veri formatları, OGC standartları çerçevesinde HTTP isteklerine (GetMap, GetCapabilities vb.), verilen yanıtlar şeklinde Web ortamında

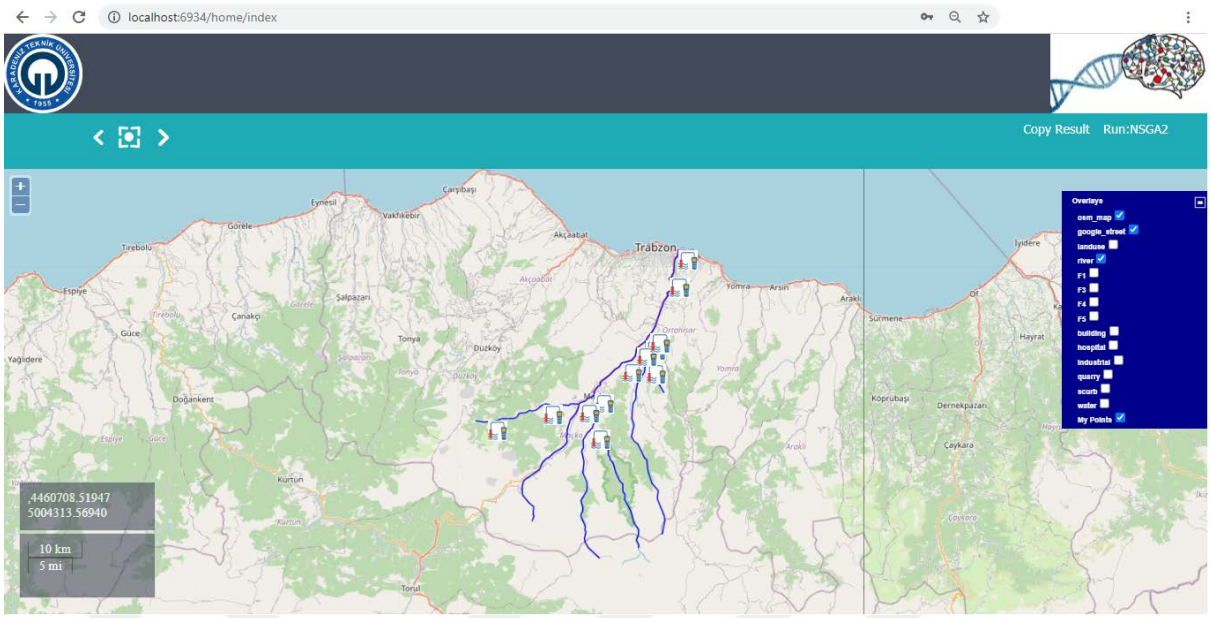
sunulabilmektedir. GeoServer, içerisinde gömülü olarak barındırdığı OpenLayers, GeoWeb Cache gibi diğer açık kaynaklı kütüphaneler ile güçlü bir harita sunucusu karakteri çizmektedir. Yazım tarihinde son sürümün desteklediği ve sunabildiği formatlar şu şekilde sıralanabilir: AtomPUB, GIF, GeoRSS, GeoTiff, JPEG, KML, OpenLayers, PDF, PNG, SVG, Tiff, CSV, GML, GeoJSON ve Shapefile.



Şekil 4.5. GeoServer yazılım görüntü ortamı

4.7.4. Web Uygulamanin Programlamasi

Bu bölümde, Visual Studio'da HTML, CSS ve JavaScript (OpenLayers) programlama dilleri yardımı ile müşteri tarafı yazılım programlaması yapıldı (Haritayı, harita koordinatlarını, harita ölçeğini vb. Görüntüleme özelliği ile). Sonra C # programlama dili ile, bu müşteri tarafı programi GeoServer programına ve PostgreSQL veritabanına bağlandı. Şekil 4.6 web tabanlı uygulamamızı göstermektedir.



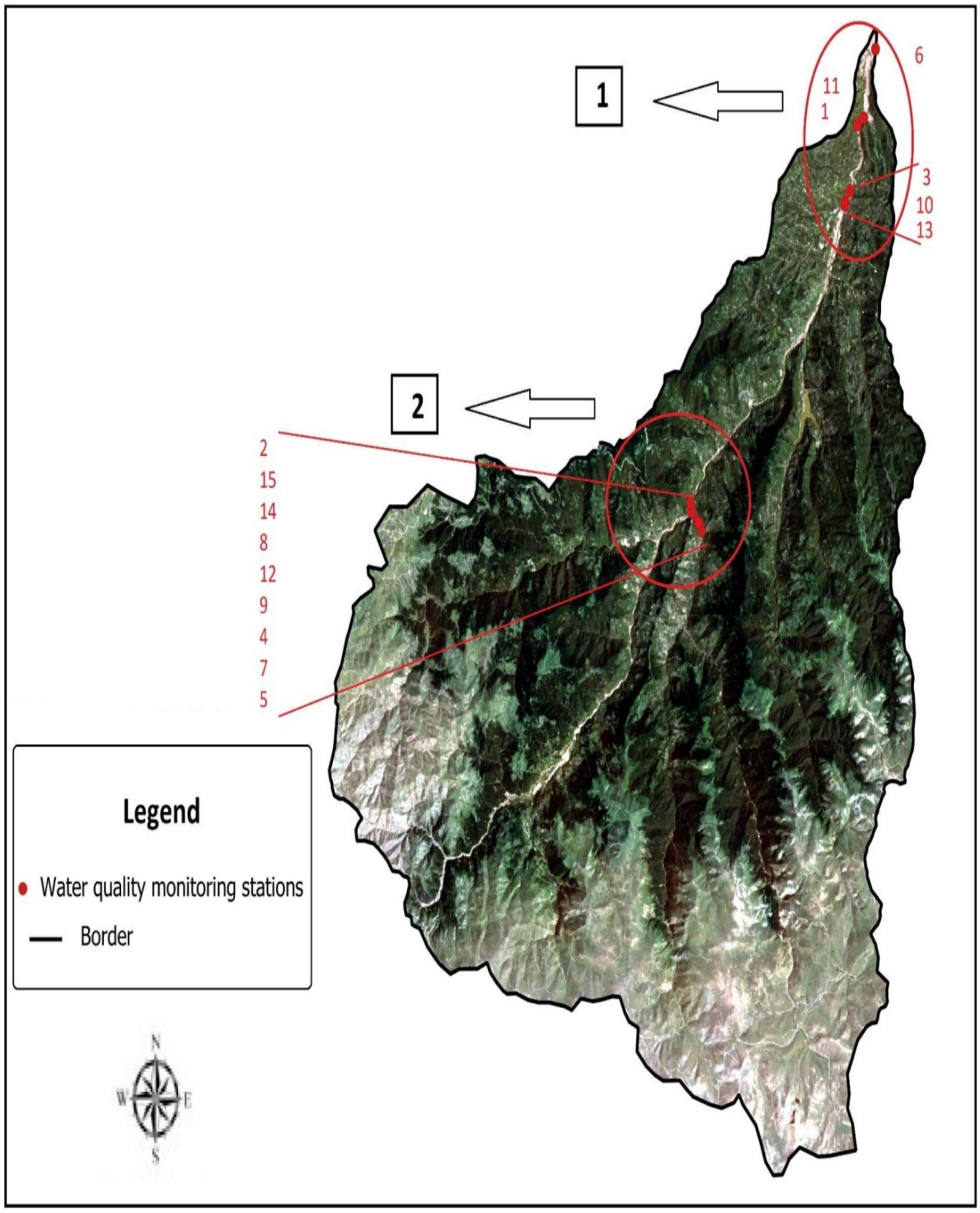
Şekil 4.6. Web tabanlı uygulama

4.7.5. NSGA-II Algoritmasının Kod Geliştirmesi

Bu çalışmada C programlama dillerindeki NSGA-II kodları kullanılmıştır (URL-2, 2021). Shapefile mekansal verilerini kullanma yeteneği eklemek için Catfood Shapefile .NET (URL-3, 2021) kitaplığı kullanıldı. Catfood.Shapefile, ESRI Shapefiles okumak için bir .NET kitaplığıdır. Sonra bu çalışmadaki optimizasyon amaçlarına göre NSGA-II algorithm kodlarındaki fitness fonksiyonları tanımlandı. Son olarak, NSGA-II algoritmasının sonuçlarını bir web haritası üzerinde görüntülemek için, NSGA-II algoritmasının çıktısı, OpenLayers kitaplığında kullanılmak üzere POI katman formatına değiştirildi (URL-4, 2021). Bir "POI", belirli bir noktayı işgal eden bir simge kullanarak belirli bir özelliği temsil etme seçeneği için haritacılıkta kullanılan bir terimdir (URL-5, 2021). OpenLayers, web tarayıcılarında harita verilerini görüntülemek için açık kaynaklı bir JavaScript kitaplığıdır.

4.7.6. NSGA-II Algoritmasının Uygulanması

Bu programda, genleri kodlamak için gerçek sayı kodlaması kullanıldı. Burada her bir kromozom, UTM koordinatlarıyla tanımlanır (örneğin:x=563667.633,y=4539303.973). Bu bölümde, NSGA-II algoritma 2500 nesil sayısı ve 20'lik ilk popülasyon sayısı,çaprazlama 0,98 ve mutasyon 0,02 ile çalıştırıldı ve su kalitesi sensörlerinin optimum yerleri elde edildi. Şekil 4.7'da Su kalitesi sensörlerinin optimum yerleri , ve Tablo 4-2'de birinci ve ikinci hedeflerin sayısal değerleri , 2500 nesil ve 20 kişilik ilk nüfus için gösterilmiştir. Çaprazlama ve mutasyon belirleme olasılıklarının genellikle deneme yanılma yöntemiyle yapıldığına dikkat edilmelidir. Literatürde çaprazlama ve mutasyon olasılıklarının değerlerini belirlemek için bir dizi kılavuz bulunmaktadır (Bäck, 1993), (De Jong, 1980). Çaprazlama olasılığının tipik değerleri 0,5~1,0 aralığındayken, mutasyon olasılığının tipik değerleri 0,001~0,05 aralığındadır (Lin vd., 2003). Çaprazlama olasılığı ne kadar yüksek olursa, GA, bir nesilden diğerine elde edilen kromozomların kalitesinin düşmemesini sağlamak için yeni kromozomlar üretmek için önceki nesilden elde edilen uygunluk puanları daha yüksek olan kromozomların daha yüksek bir olasılığını kullanır. Mutasyon olasılığı, GA'ların yeni bir alanı keşfetme hızını kontrol eder. Mutasyon, arama sürecinin yerel optimal tuzaklardan kaçmasına yardımcı olmak için ek kromozom çeşitliliğine yol açar. Çaprazlama olasılığı ne kadar yüksek olursa, sömürü o kadar hızlı gerçekleşir. Çok büyük bir çaprazlama olasılığı, bireyleri sömürü edilebileceklerinden daha hızlı bozar. Mutasyon olasılığı, GA'ların yeni bir alanı keşfetme hızını kontrol eder.



Şekil 4.7. Su kalitesi sensörlerinin optimum yerleri (ilk nüfus 20 ve nesil sayısı 2500).

Tablo 4.2. Birinci ve ikinci hedeflerin sayısal değerleri(ilk nüfus 20 ve nesil sayısı 2500)

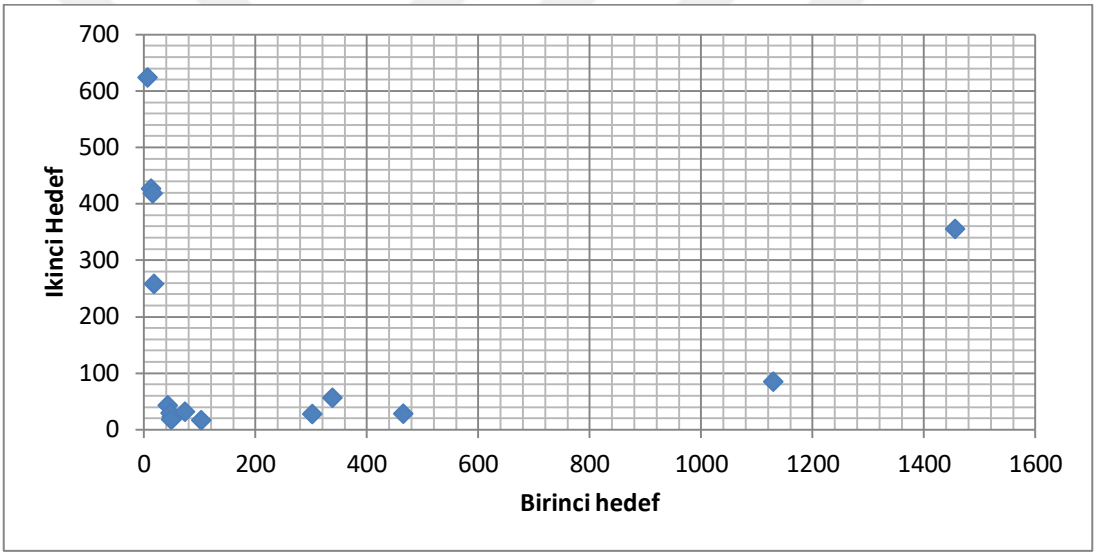
	lat	lon	Birinci hedef (F1)	İkinci hedef (F2)
1	40/96561	39/74172	56/82805668	37/63994683
2	40/81532	39/61032	48/05827702	21/48617759
3	40/93935	39/73565	88/83093241	14/002812
4	40/80685	39/61713	4/609223067	221/001317
5	40/80414	39/61872	4/938050862	262/1776499
6	40/99587	39/75615	4/180466038	10/32816622
7	40/80621	39/61767	1/602575285	130/569411
8	40/80821	39/61576	30/28773708	31/91595303
9	40/80735	39/6167	6/696607638	594/3765469
10	40/93514	39/73205	42/43264109	34/94630908
11	40/96849	39/74633	15/83326728	8/364155714
12	40/80737	39/61668	6/862211829	517/804754
13	40/93385	39/73134	53/63728407	24/98543921
14	40/8118	39/61118	9/927552169	10/70213537
15	40/81532	39/61032	48/05827702	21/48617759

Şekil 4.7'de gösterildiği gibi, iki alanda su kalitesi izleme istasyonlarının birikmesi gözlemlenmektedir. Çalışma alanının görsel olarak incelenmesiyle, Bölge 1'de istasyonların birikmesinin nedeninin atık su üretimi nedeniyle nehir suyu kirliliği üzerinde büyük etkisi olabilecek endüstrilerin varlığından kaynaklandığı görülmüştür. Bölge 2'de istasyonların birikmesinin nedeni, bu alanlardaki hasarlı orman alanlarının büyüklüğünden dolayı tortu üretimine neden olmakta ve nehrin su kalitesini etkilemektedir. Diğer alanlarda, nehir suyu kalitesini etkileyen faktörlerin olmaması ve iletişim tesislerinden uzaklık nedeniyle, GA tarafından su kalite istasyonları için uygun bir yer bulunamamıştır.

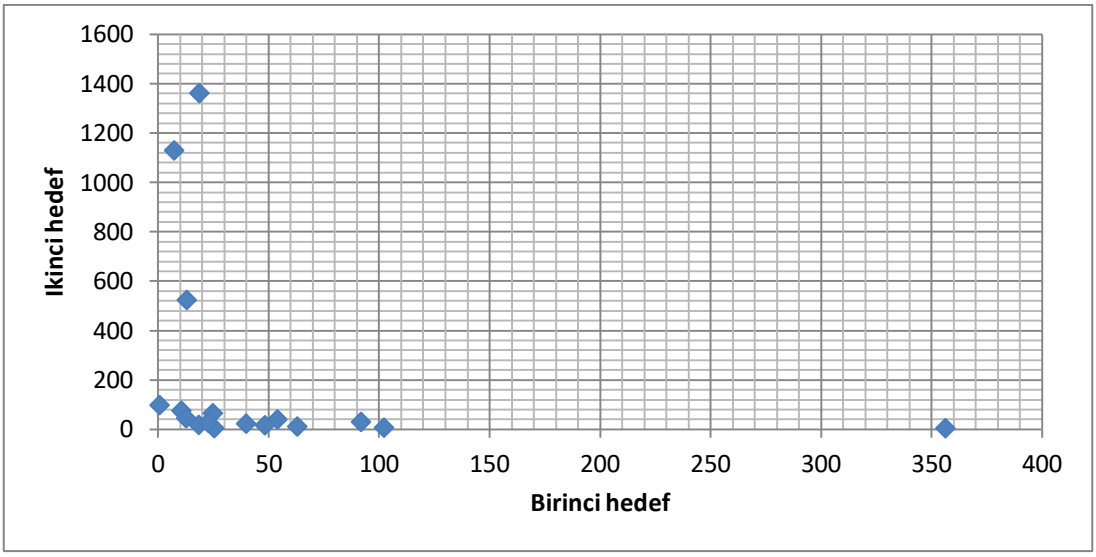
4.8. NSGA-II Algoritmasının Testi ve Sonuçlarının Karşılaştırılması

4.8.1. Optimal Pareto Çözümlerinin Araştırılması

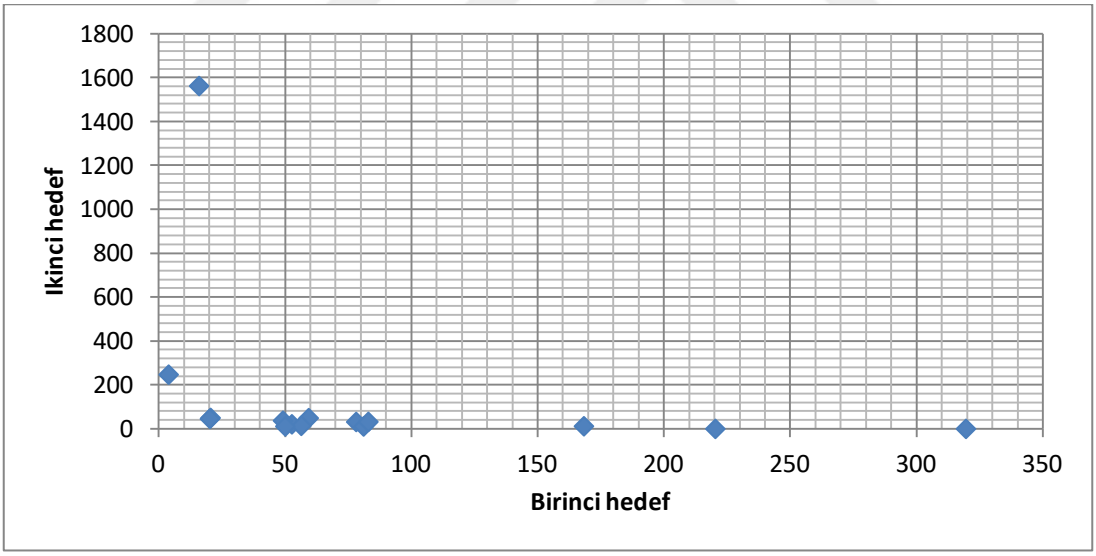
Bu bölümde, NSGA-II algoritma 500,1000,1500, 2000, 2500 nesil sayısı ve 20'lik ilk popülasyon sayısı,çaprazlama 0,98 ve mutasyon 0,02 ile beş kez çalıştırıldı. Pareto optimal çözüm diyagramları Şekil 4.8 ila 4.12'da gösterilmiştir.



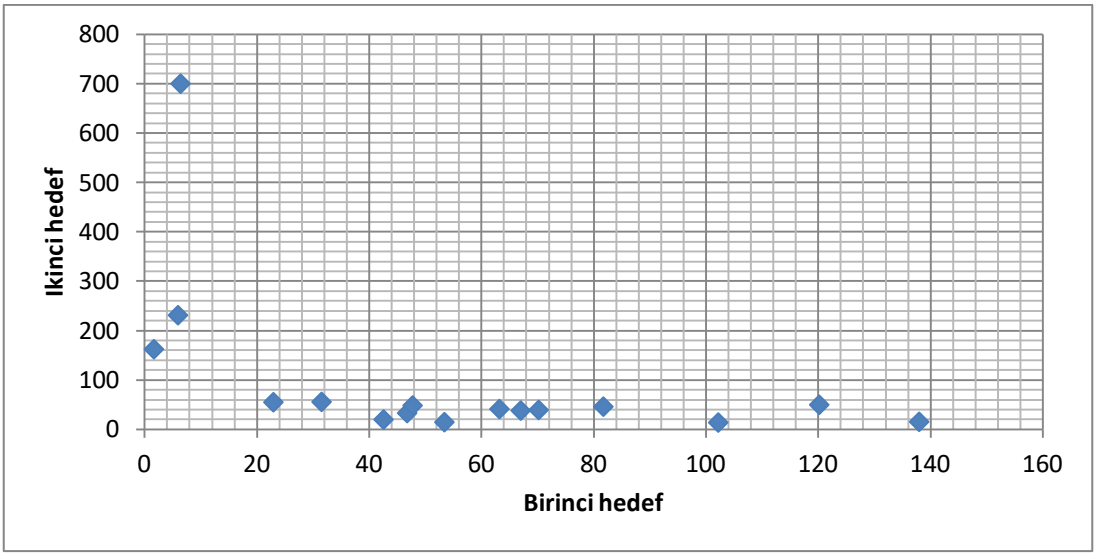
Şekil 4.8. İlk hedef ile ikinci hedef arasındaki Pareto optimal çözümleri grafiği (ilk nüfus 20 ve nesil sayısı 500)



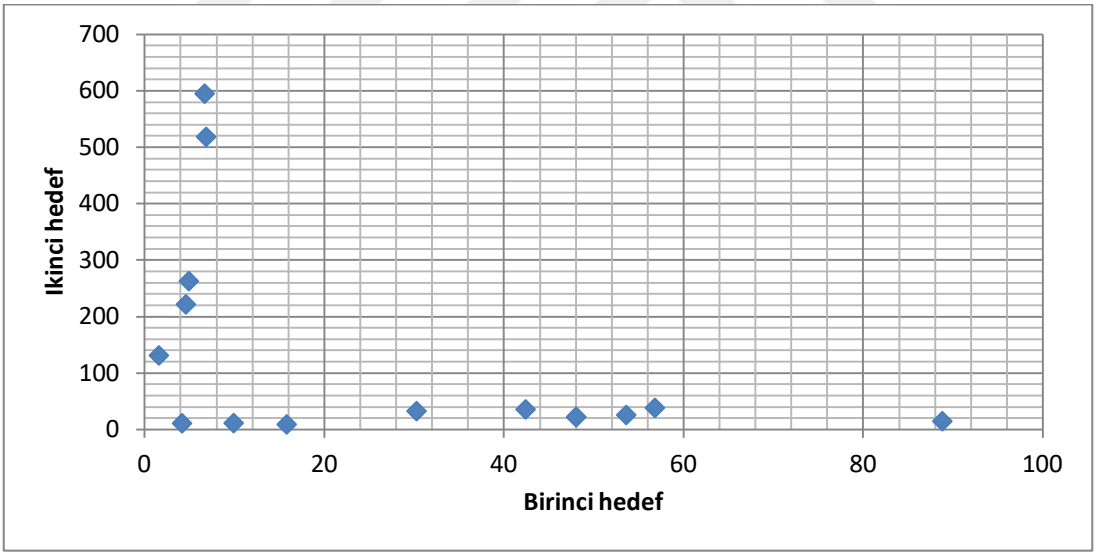
Şekil 4.9. İlk hedef ile ikinci hedef arasındaki Pareto optimal çözümleri grafiği (ilk nüfus 20 ve nesil sayısı 1000)



Şekil 4.10. İlk hedef ile ikinci hedef arasındaki Pareto optimal çözümleri grafiği (ilk nüfus 20 ve nesil sayısı 1500)



Şekil 4.11. İlk hedef ile ikinci hedef arasındaki Pareto optimal çözümleri grafiği (ilk nüfus20 ve nesil sayısı 2000)

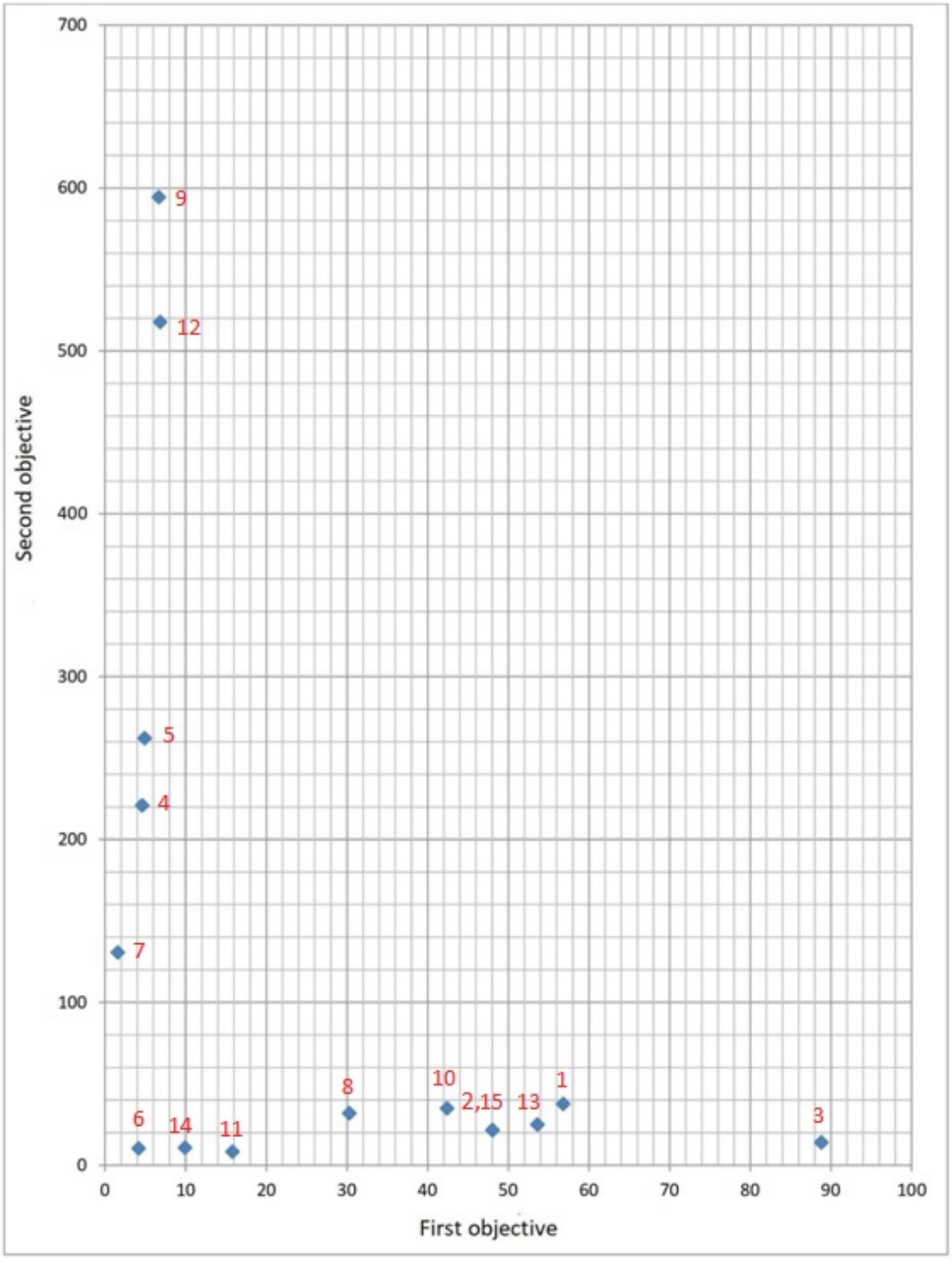


Şekil 4.12. İlk hedef ile ikinci hedef arasındaki Pareto optimal çözümleri grafiği (ilk nüfus 20 ve nesil sayısı 2500)

Tablo 4.3. algoritma çalıştırma süresi

Nesillerin sayisi	Populasyonun sayisi	Zaman(Dakika)
500	20	34:19:00
1000	20	69:55:00
1500	20	107:21:00
2000	20	143:27:00
2500	20	179:18:00

Şekil 4.8 ila 4.12'te gösterildiği gibi, nesil sayısı 500'den 2500'e çıktıkça birinci ve ikinci hedeflerin sayısal değerleri azalmaktadır, ancak nesil sayısının 3.000'e yükselmesiyle, NSGA-II algoritmasının yürütme süresinde 36 dakikalık bir artış olmasına rağmen, birinci ve ikinci hedeflerin sayısal değerlerinin çoğunun azaltılmasında önemli bir fark yoktu. Birinci hedefin en yüksek değeri 88.83'ten 87.6'ya, ikinci hedefin en yüksek değeri 594.37'den 594.2'ye düştü. Sonuç olarak, 2500 neslinden elde edilen optimal cephe nihai sonuç olarak seçildi. Şekil 4.14, ilk cephede 2500 nesil NSGA-II algoritmadan elde edilen Pareto optimal çözümlerini gösterir.



Şekil 4.13. İlk hedef ile ikinci hedef arasındaki Pareto optimal çözümleri grafiği (ilk nüfu 20 ve nesil sayısı 2500)

Pareto optimal çözümlerinden nihai çözümlerin seçiminde, bu çözümlerin hem Pareto optimal cephesindeki konumu hem de çalışma alanında aynı anda düşünülmelidir. Pareto Optimal Cephede, bu cephenin ortasındaki çözümler, Pareto sınır eğrisinin iki ucundaki çözümlere veya ekstrem çözümlere kıyasla iki hedef arasında daha iyi bir dengeye sahiptilar. Sonuç olarak, bu çözümlere nihai çözümler olarak seçimde öncelik verilir. Ekstrem çözümler, her bir amacın minimum veya maksimum değerine sahip Pareto çözümleridir. Diğer bir deyişle, bir optimizasyon probleminde ekstrem noktalar, Pareto sınır eğrisinin uç noktalarıdır (Bai vd., 2012). Nihai çözümleri seçmek için, çözümlerin çalışma alanındaki konumu da karar verici tarafından dikkate alınmalıdır. Çözümün tüm çalışma alanında dağılımını sağlamak için öncelik, birbirinden daha uzak olan çözümlerin seçilmesidir. Hem Pareto optimal cephesinde hem de çalışma alanında çözümlerin birbirine yakın olması durumunda karar verici çözümlerden birini nihai çözüm olarak seçebilir.

Pareto optimal cephe incelemesinde, dikey eksende optimal Pareto cephesinin üst kısmının (5, 9, 12 noktaları) düz bir çizgi olduğu görülebilir, ve ikinci amaç fonksiyonunun sayısal değerleri azaldıkça, birinci amaç fonksiyonunun sayısal değerleri sabit kalır ve değişmez. Araştırma literatürünün gözden geçirilmesi, bunun, NSGA-II algoritması da dahil olmak üzere MOO algoritmaları, Pareto optimal çözümler yerine zayıf Pareto optimal çözümleri bulduğunda meydana geldiğini göstermektedir. Pareto optimal çözümlerinin, zayıf Pareto optimal çözümlerin bir alt kümesi olduğunu belirtmekte fayda var (Miettinen, 1998). Resmi olarak Pareto optimal çözümler ve Zayıf Pareto optimal çözümlerin tanımı aşağıdaki gibidir:

Pareto optimal çözümlerinin tanımı:

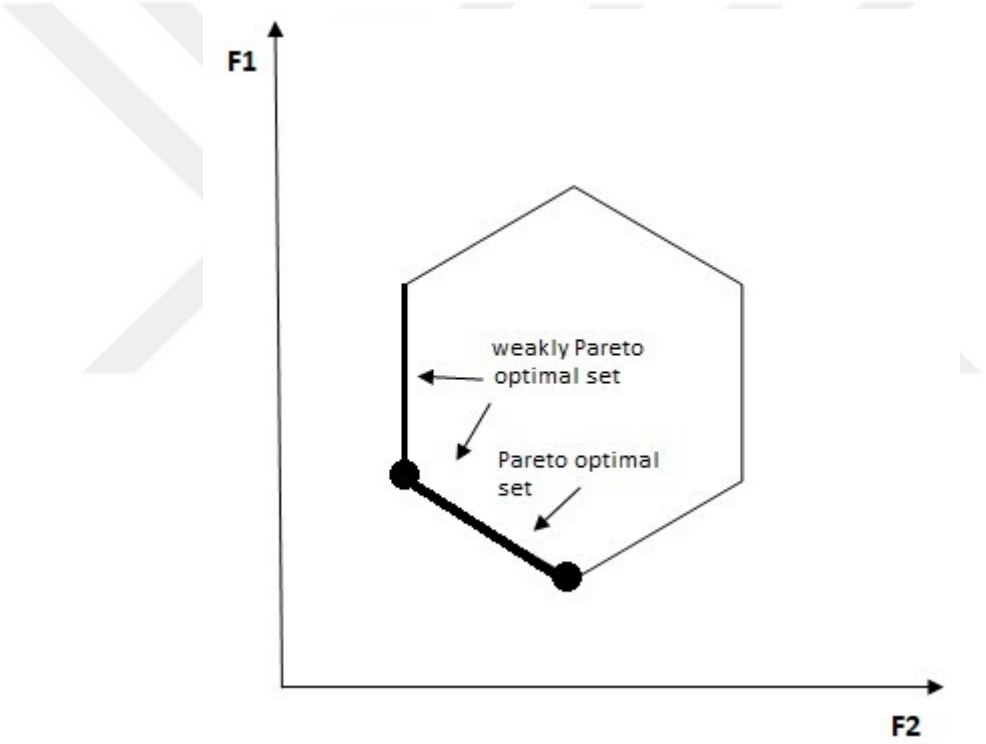
Bir karar vektörü X* ∈ S, Pareto optimaldir, eğer $fi\ (X^*) \leq fi\ (X)$, karar vektörü X^* her bir amaç fonksiyonunda X kadar iyi bir değer vermelidir, ve $fj\ (X^*) < fj\ (X)$, Karar vektörü X^* en az bir amaç fonksiyonunda kesinlikle X'den daha iyi performans göstermelidir (Miettinen, 1998).

Zayıf Pareto optimal çözümlerinin tanımı:

Bir karar vektörü X* ∈ S zayıf Pareto optimaldir, eğer fj (X*) < fj (X), karar vektörü X* en az bir amaç fonksiyonunda kesinlikle X'den daha iyi performans göstermelidir (Miettinen, 1998).

"x", "R^n" karar değişkeni uzayının bir alt kümesi olan (boş olmayan) uygun bölge "S"ye ait karar (değişken) vektörüdür.

Şekil 4.14'teki kalın çizgi, zayıf Pareto optimal hedef vektörleri kümesini temsil eder. Pareto optimal kümesinin zayıf Pareto optimal kümesinin bir alt kümesi olduğu da şekilde görülebilir. Pareto optimal amaç vektörleri, noktalar arasındaki çizgi boyunca yer alır (Miettinen, 1998).



Şekil 4.14. Zayıf Pareto optimal vektörler (Miettinen, 1998).

Zayıf Pareto optimal çözümlerinin (çözüm 5, 9, 12) konumu incelendiğinde, bu çözümlerin Şekil 4.7'de belirtilen bölge 2'de ve diğer çözümlerin yanında yer aldığı görülmektedir. Sonuç olarak, bu çözümlerin kaldırılması, çalışma alanındaki çözümlerin dağılımını azaltmaz. Bu zayıf Pareto optimal çözümlerin bir önceki paragrafta anlatılan

ekstrem çözümler arasında olduğu da gözlemlenebilir. Yukarıda belirtilen içeriklere gore 1, 2, 4, 6, 7, 8, 10, 11, 13,14,15 çözümleri nihai çözüm olarak seçilebilir.

4.8.2. NSGA-II ile Analitik Hiyerarşik Süreç[1] Sonuçlarının Karşılaştırılması

GA'nın sonuçlarını doğrulamak için iki yöntem kullanıldı: 1- Hiyerarşik analitik süreç yöntemiyle yer seçimi 2- Bulanık mantık yöntemiyle yer seçimi. Her iki yöntem de aşağıda açıklanmıştır.

4.8.2.1. AHP Yöntemiyle Yer Seçimi

4.8.2.1.1. Analitik Hiyerarşik Süreç

1970'li yıllarda Thomas Saaty tarafından geliştirilen Analitik Hiyerarşik süreç (AHP) birden fazla ölçüt içeren karmaşık problemlerin çözümünde kullanılan çok ölçütlü bir karar verme yöntemidir (Kuruüzüm ve Atsan, 2001). AHP, karmaşık problemleri amac ölçütler alt ölçütler seçenekler hiyerarşisi kurularak çözmeye olanak sağlamaktadır. AHP, genel olarak, problemi parçalara ayırma ve hiyerarşi oluşturma[2]; karşılaştırmalı karar verme ve tercih matrisinin oluşturulması[3]; ve önceliklerin sentezlenmesi[4] olmak üzere üç temel adıma dayanmaktadır (Saaty, 1980).

AHP geliştirildiği günden beri birçok alanda uygulama olanağı bulmuştur. Yer analizi (Min, 1994), kaynak tahsisi (Cheng ve Li, 2001), pazarlama (Davies, 2001), enerji (Kim ve Min, 2004), eğitim (Saaty, 1991a), risk analizi (Millet ve Wedley, 2002), çevresel etki değerlendirme (Ramanathan, 2001) ve arazi uygunluk analizi (Banai-Kashani, 1989) bu uygulama alanlarından bazılarıdır.

---

[1] Analytical Hierarchy process
[2] Decomposition
[3] Comparative judgement
[4] Synthesis of priorities

4.8.2.1.2. Yer Seçiminde AHP

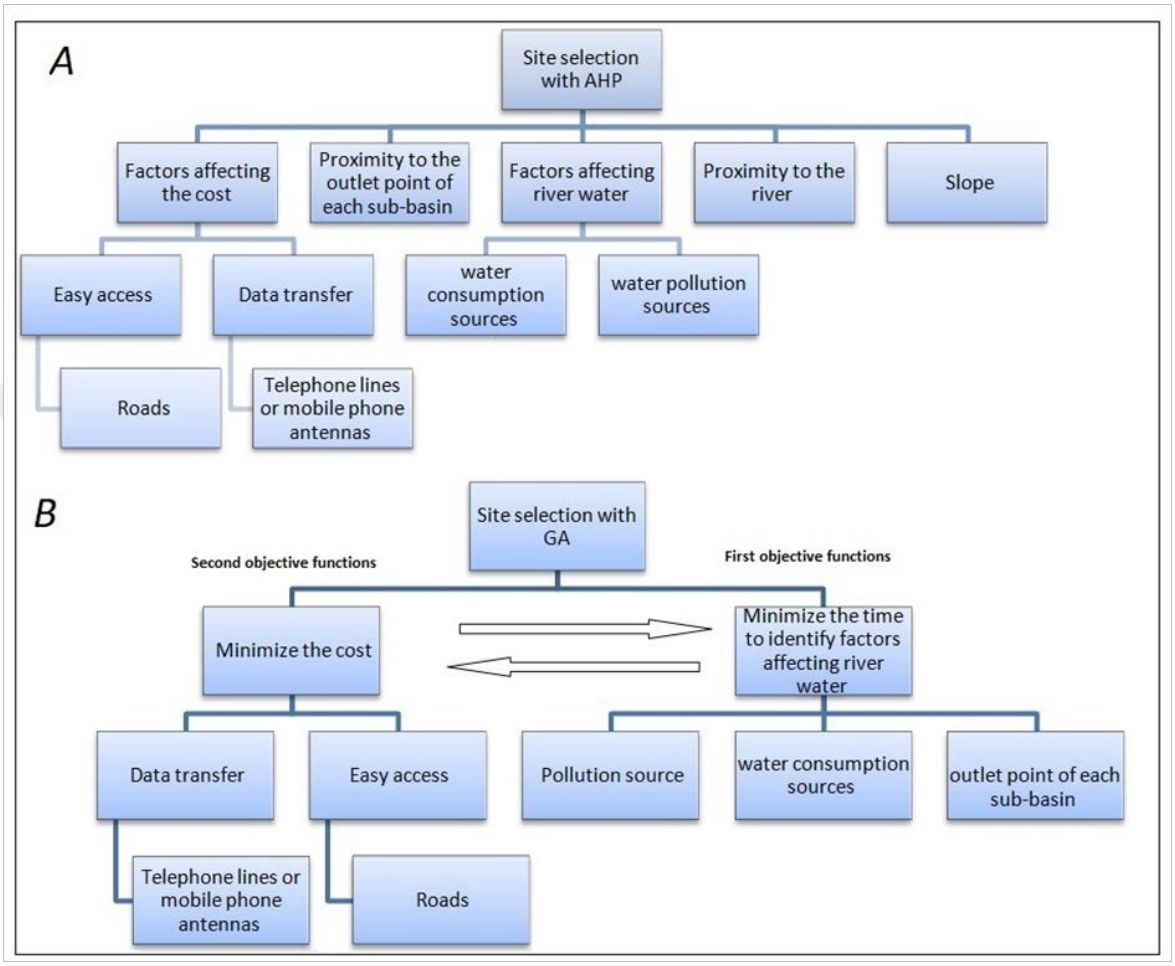
Yer seçimi, tanımlanan bir arazi kullanımı için uygun bir alanın belirlenmesinde kapsamlı bir biçimde saptanmış amaç ve faktörlerin birlikte düşünülmesini gerektirmektedir. Örneğin endüstriyel bir tesis için yer seçimi işlemi ekonomik, sosyal, teknik ve çevresel disiplinlerce tanımlanmış bir dizi faktörü içermektedir. Özellikle tüm bu faktörlerin bir yer seçimi için birlikte düşünülmesi bu işlemi oldukça karmaşık hale getirmektedir (Jun, 2000).

Yer seçimi problemi sonuç ürün olan karar vermede anahtar rol oynayan ölçütleri içeren bir çok ölçütlü karar verme problemidir. Bir tercih modeli oluşturularak karar vericilerin tercihlerinin değerlendirilmesi amacıyla birçok kuram ve yöntem geliştirilmiştir. Literatürde etkin bir yaklaşım olarak AHP uygulama alanı bulmaktadır (Saaty, 1994). Yer seçimi eylemi öncelikle bir yer seçimi kararının alınmasıyla başlamaktadır. Sonraki adımda yer seçimi faktörleri tanımlanmaktadır. İlgili faktörlerin ağırlıkları belirlendikten sonra her bir olası yerleşimin sıralanması başarılmaktadır. Karşılaştırma sonuçları analiz edilerek tercih edilen yerleşimlerin tanımlanması ve en uygun yerleşimin önermesiyle işlem sonlanmaktadır.

Yer seçimi analizinde AHP, endüstri bölgelerinin seçiminden (Eldrandaly ve diğ, 2003); katı atık depoları için en uygun yer seçimine (Siddigui ve diğ, 1996) kadar geniş bir alanda uygulama alanı bulmaktadır. Değinilen AHP destekli yer seçimi uygulamalarına ek olarak fuar alanı seçimi (Chen, 2006), hastane yeri seçimi (Ohta vd., 2007) ve mağaza yeri (Burnaz ve Topçu, 2006) de verilebilir.

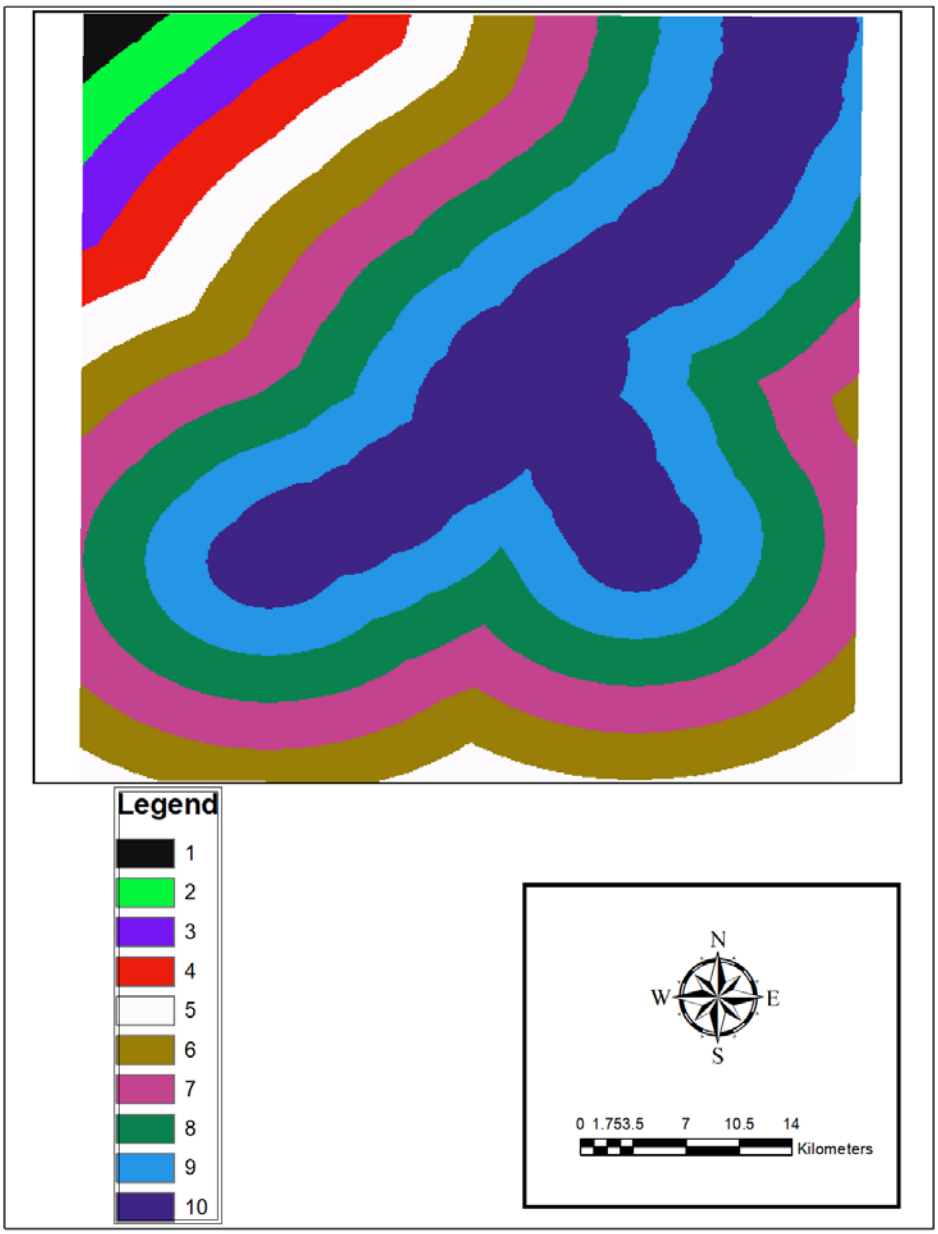
AHP yöntemi ile yer seçimine başlamak için öncelikle nehirdeki su kalite istasyonları için yer seçimi için etkin kriterler belirlenmelidir. Burada GA kullanılan aynı kriterler kullanıldı. Bu kriterler, 1- Su kalitesi değişikliklerini etkileyen dış nedenlerin ve kaynakların belirlenmesi için nehir suyu kirlilik kaynakları 2- Su kaynaklarının kullanımını desteklemek için nehir suyu tüketim kaynakları 3- Belirli bir dönemde amaçlı araştırmalar yoluyla su kalitesinde kısa vadeli değişiklikleri incelemek ve her havzadan kirlilik yüklerini tahmin etmek için çalışma alanındaki her bir alt havzanın çıkış noktasına yakınlık 4- Su kalitesi izleme istasyonlarına kolay ve güvenli erişim için yollar 5- Sensörlerde kayıtlı bilgileri iletmek için telefon hatları veya cep telefonu antenleri 6- Eğim, eğim ne

kadar düşükse, su kalitesi izleme istasyonu inşa etme maliyeti o kadar düşük olur 7- Nehire yakınlık (çünkü istasyonlar nehir üzerinde olmalıdır). GA ve AHP yöntemlerinde kullanılan kriterler Şekil 4.15'te gösterilmiştir.

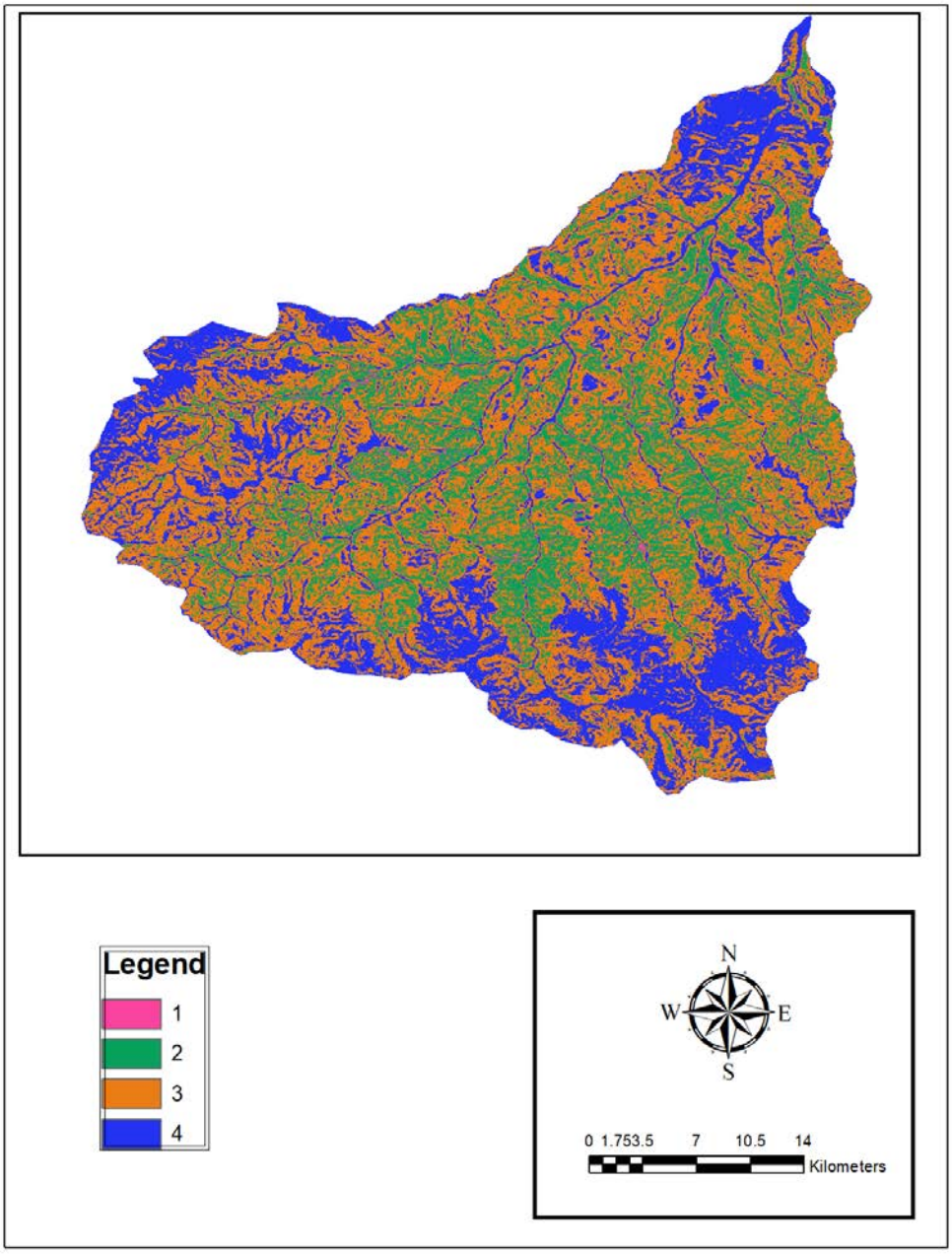


Şekil 4.15. Kullanılan kriterler A) AHP yöntemi B) GA yöntemi.

Bir sonraki adımda, bu kriterlerin her birinin mesafe haritası Arcgıs yazılımında oluşturulmuş ve ardından 1'den 5'e kadar olan kriterlerin mesafe haritalarının her biri 10 sınıfta yeniden sınıflandırılırlar. Çalışma alanında en yakın yere 10 (daha yüksek değer) ve en uzak yere 1 (düşük değer) puan verildi. Ayrıca alanın eğim haritası 4 sınıfa ayrılmıştır. En düşük eğime (0-15 derece) 4 (daha yüksek değer) puan ve 1 puan (daha düşük değer) (45 derece ve üstü) verildi. Örneğin, Şekil 4-16 ve 4-17'te yolların yeniden sınıflandırılmış mesafe haritasını ve yeniden sınıflandırılmış eğim haritasını gösterilmiş.



Şekil 4.16. Yolların yeniden sınıflandırılmış mesafe haritası



Şekil 4.17. Yeniden sınıflandırılmış eğim haritasını

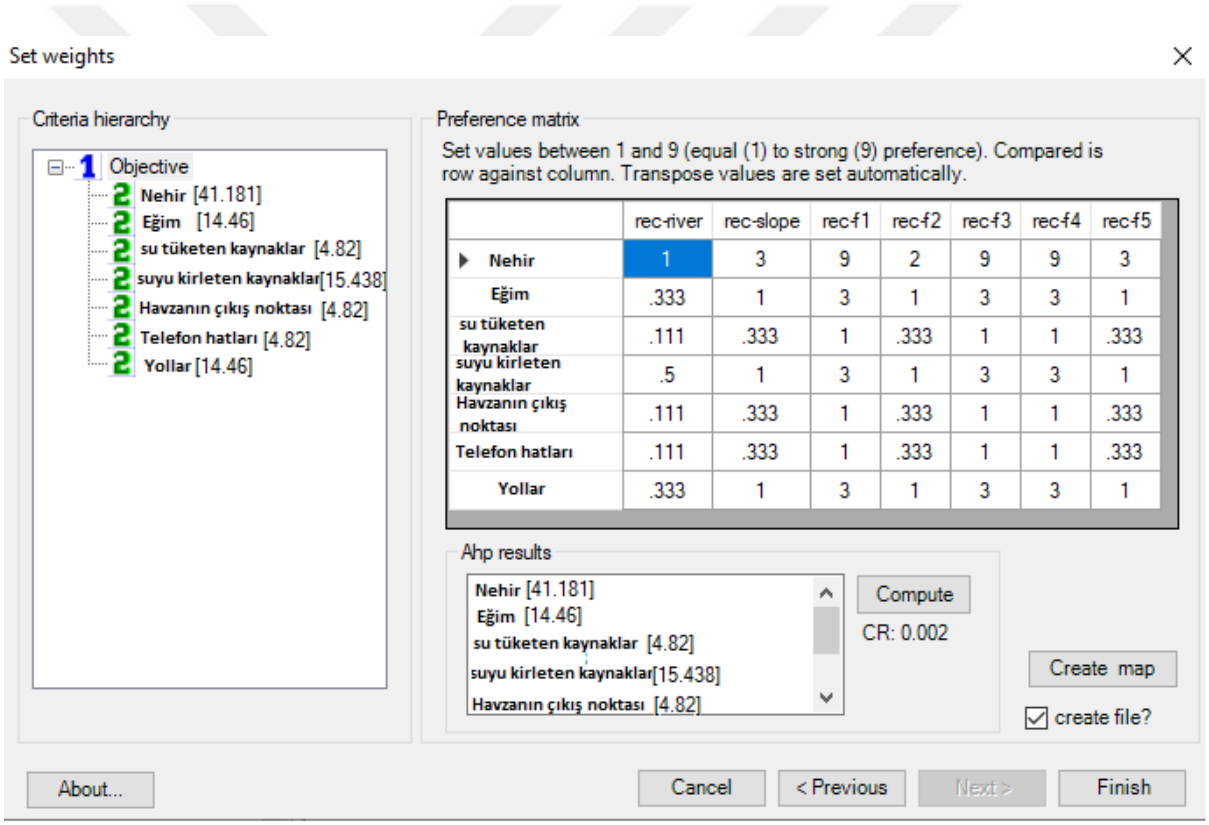
Sonraki adımda yer seçiminde etkili olan kriterlerin her biri, hiyerarşik analiz süreci ile ve uzmanların görüşüne dayalı çift olarak karşılaştırıldı ve yer seçiminde etkili katmanların her birinin ağırlığı elde edilmiştir (Şekil 4.18) ve son harita, ArcGIS'teki AHP

extension piksel değerlerinin ağırlıklı toplam yöntemi (Denklem 4.1) ile elde edildi. (Şekil 4.19).

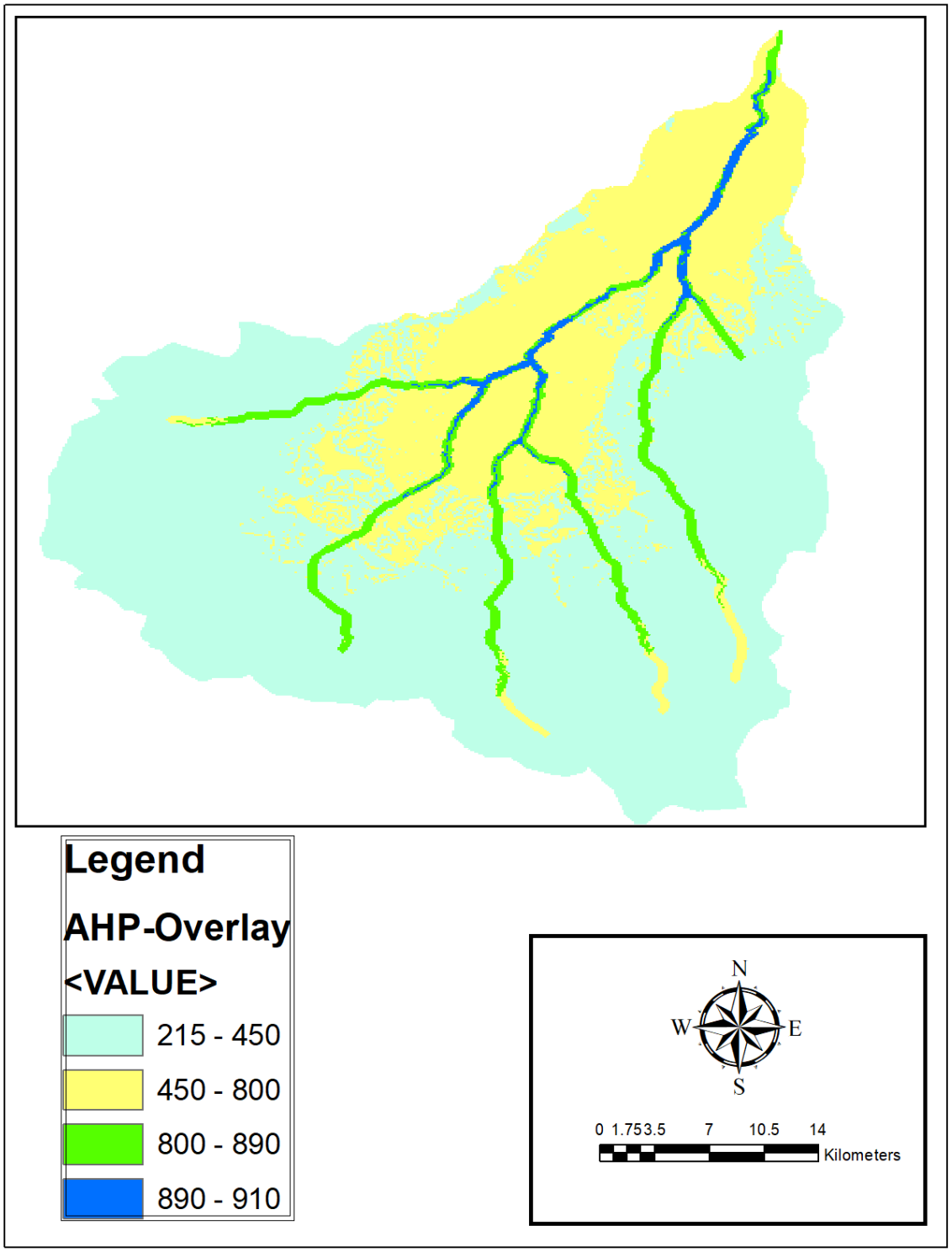
$$si = \sum_{j=1}^{m}(eji * wj) \quad (4.1)$$

burada "s_i", "i" alternatifinin değerlendirme puanını, "e_{ji}", "j" kriterinin "i" alternatifine göre kriter puanını, "w_i", tüm kriterler arasında bir kriterin önemini gösteren kriter ağırlıklarını ifade eder. Ağırlık vektörünün ağırlık değerlerinin toplamının "1"e eşit olduğuna dikkat edilmelidir. Yani,

$$\sum_{j=1}^{m} wj = 1 \quad (4.2)$$

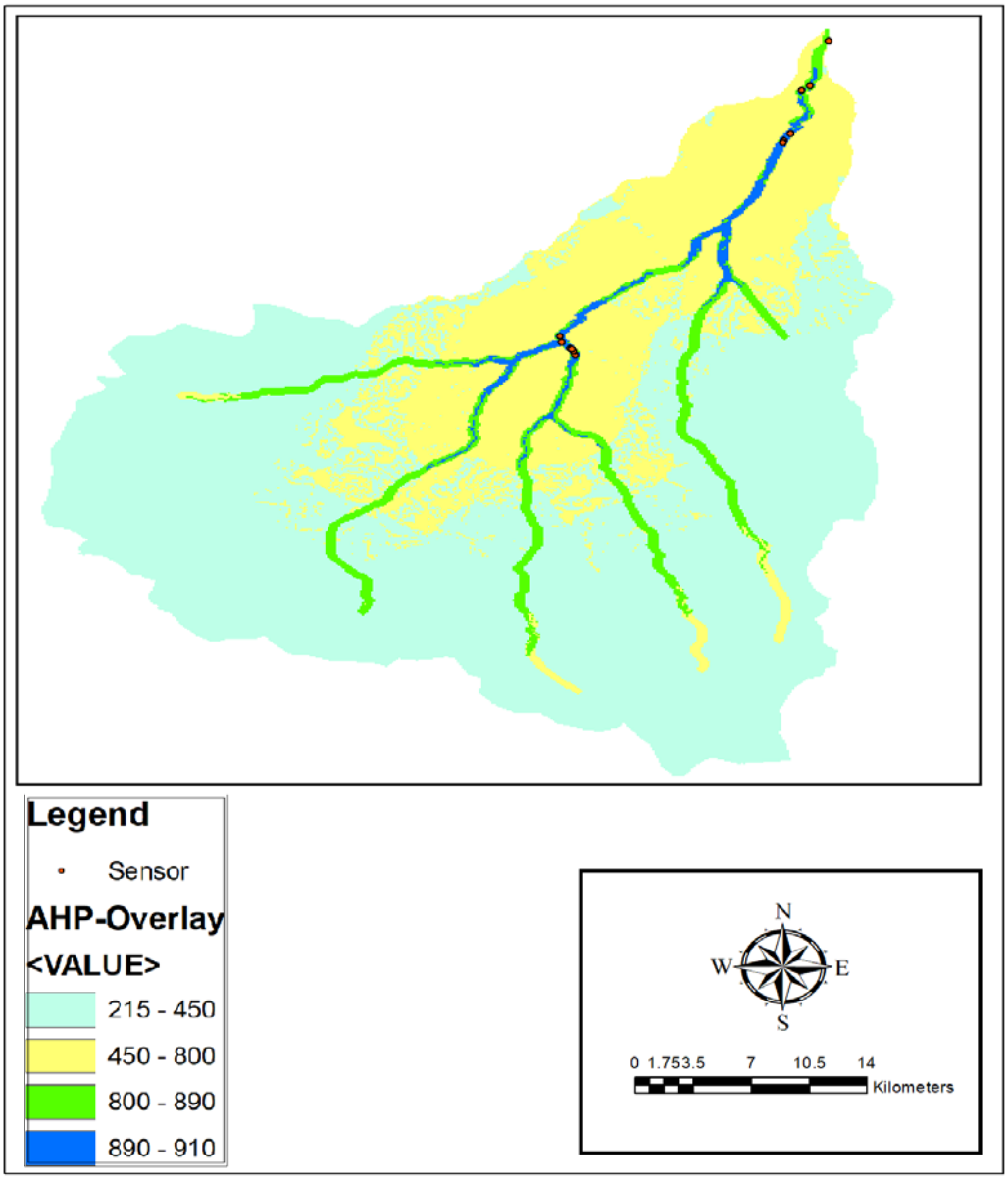


Şekil 4.18. kriterlerin çift karşılaştırılması ve ağırlıkları



Şekil 4.19. AHP yöntemiyle su kalitesi sensörleri için uygun yerler

Şekil 4.19'te gösterildi gibi, koyu mavi renkli yerler (910-890 değerleri) su kalitesi sensörlerini kurmak için çok uygun yerlerdir ve yeşil renkli yerler (890-800 değerinde) su kalitesi sensörleri kurmak için orta derecede uygundur ve sarı renkli yerler (450-800 değerinde) su kalitesi sensörleri kurmak için daha az tercih edilir. Şekil 4.19, NSGA-II algoritma yöntemi ve hiyerarşik analitik sürecinin sonuçlarının aynı anda gösterilmesinden elde edilen haritayı göstermektedir.



Şekil 4.20. NSGA-II algoritması ile AHP sonuçlarının karşılaştırılması.

Şekil 4.20'te gösterildi gibi, NSGA-II algoritma yönteminden (kırmızı noktalar) elde edilen 15 sensör lokasyonundan 13 sensör lokasyonu hiyerarşik analitik süreci yönteminden elde edilen çok uygun lokasyonlarda yer almaktadır (Koyu mavi ile gösterilen alanlar). Bu sonuçlar, yukarıdaki iki yöntemin% 86 uyumlu olduğunu göstermektedir.

AHP bir karar verme yöntemidir. İlhamı, insan zihninin karmaşıklıkla başa çıkmak için hiyerarşik ayrıştırmayı kullanması gerçeğinden gelir. AHP'nin amacı, ikili karşılaştırmalar yoluyla kriterler arasında oran ölçekleri elde etmektir. Bu karşılaştırmalar, gerçek ölçümlerden veya kriterlerin göreli gücünü yansıtan temel bir ölçekten alınabilir (Saaty, 2001). AHP, üç temel adıma dayanmaktadır: problemi parçalamak ve bir hiyerarşi oluşturmak; karşılaştırmalı karar verme ve tercih matrisi oluşturma ve göreli ağırlıkları elde etme. Ancak bölüm 2.2'de bahsedildiği gibi, GA'lar, doğal seçilim ve genetik ilkelerine dayanan arama ve optimizasyon algoritmalarıdır. AHP yönteminden farklı olarak, çözümlerin amaç fonksiyonu tarafından doğrudan değerlendirilmesi, uzman görüşüne dayalı kriterlerin ikili karşılaştırmalarına gerek kalmadan yinelemeli ve otomatik bir süreçtedir. İki yöntem arasındaki ilk fark, kriterlerin veya hedeflerin eklenmesi veya çıkarılmasıdır. AHP yönteminde amaç, kriter veya alt kriterlerdeki herhangi bir değişiklik AHP'nin hiyerarşik yapısına uygulanmalı ve bu hiyerarşik yapı değiştirilmeli, kriter ve alt kriterlerin ikili karşılaştırmasının tüm adımları tekrarlanmalıdır. Ancak GA yönteminde amaç ve kriterlerin değişimi, GA'nın yapısını değiştirmeye gerek kalmadan sadece amaç fonksiyonlarına uygulanır. Amaç fonksiyonu, verilen bir çözümün istenen problemin optimum çözümüne ne kadar yakın olduğunu değerlendiren fonksiyondur. İki yöntem arasındaki ikinci fark, AHP'nin hiyerarşik bir yapıya sahip olmasından kaynaklanmaktadır. Temel AHP'nin hiyerarşik yapısı, kriterler arasındaki bağımlılıkların yalnızca hiyerarşinin seviyeleri arasında olmasına izin verir ve olası tek etki yönü hiyerarşinin en üstüne doğrudur. Bu, modele geri besleme ilişkilerini dahil etme olasılığını ortadan kaldırır. Ayrıca, belirli bir düzeyin kriterlerinin karşılıklı olarak bağımsız olduğu varsayılır (Hamalainen ve Seppalainen, 1986). GA'da doğrusal bir hiyerarşik yapı oluşturmaya gerek yoktur ve belirli bir seviyede mevcut olan kriterler veya hedefler bağımsız değildir ve birbirleriyle geri bildirim ilişkisine sahip olabilir. Çünkü çok amaçlı GA da dahil olmak üzere çok amaçlı optimizasyon problemlerinde, çelişen amaçlara sahibiz ve bir amaç

fonksiyonundaki herhangi bir iyileştirmeye, başka bir amaç fonksiyonundaki kötüleştirir (Gen, vd., 2008). Bu, bu hedeflerin bağımsız olmadığını ve birbirleriyle geri bildirim ilişkileri olduğunu göstermektedir. Ayrıca AHP, çok boyutlu bir problemi tek boyutlu bir probleme indirger. Kararlar, en iyi sonuç için tek bir sayı ile veya farklı olası sonuçların sıralamasını veren bir öncelikler vektörü ile belirlenir (Saaty, 2013). GA'da da, benzer kriterleri ve hedefleri birleştirerek ve gruplayarak, çok boyutlu bir problem azaltılabilir ve Daha Az boyutlu bir probleme dönüştürülebilir, ancak mutlaka tek boyutlu bir probleme dönüşmez. Ahmedi Pari vd., (2018), endüstriyel alanların yer seçimine yönelik bir çalışmada, GA ve bulanık-AHP olmak üzere iki yöntem kullanmış ve bu iki yöntemin sonuçlarını duyarlılık analizi ile her bir kriterin uygun konumları ile karşılaştırmışlar. GA sonuçlarının, bulanık-AHP yöntemine kıyasla girdi katmanları ile daha iyi örtüştüğünü gözlemlemişler ve GA ile yer seçiminin, yalnızca yaklaşık sonuçlar gösterebilen bulanık-AHP yöntemine göre daha doğru ve verimli sonuçlar verdiği sonucuna varmışlardır. Bir başka çalışmada Peng ve Li (2015) kablosuz sensör ağların yerini belirlemek için bir GA kullanmış ve sonuçları simüle ederek, GA'nın kablosuz sensör ağların yerini belirlemede kullanılan diğer yöntemlere göre hem konum doğruluğu hem de yakınsama hızında daha iyi olduğunu göstermişler. Başka bir çalışmada Chicco ve Mazza (2020), GA dahil evrimsel algoritmaların, sonuçların otomatik olarak belirlenmesi ve farklı yeterlilik seviyelerine sahip farklı uzmanların görüşlerinden etkilenmemesi gereken teknik konularda daha iyi bir seçim olduğunu öne sürmüşler.

4.8.2.1. NSGA-II ile Bulanık Mantık[1] Sonuçlarının Karşılaştırılması

4.8.2.1.1. Bulanık Mantık

Bulanık mantık ilk olarak modern hesaplamalarda, Lotfi A Zadeh (1966) tarafından bulanık kümeler teorisini hazırlayarak ortaya çıktı. Bulanık terimi, yanlış, belirsiz (floating) anlamına gelir. Bu teorinin avantajı, tek bir sayı yerine, katılım olasılığı derecelerine sahip bir sayılar alanı dikkate alınmıştır. Bu nedenle, faktörlerin ağırlığını ifade etmek için bulanık kümeler teorisi kullanıldığında, kesinlikle tek bir sayı ağırlık

[1] Fuzzy logic

olarak belirlenmeyecektir, ancak yalnızca önerilen sayının doğruluk olasılığı diğer yakın sayılardan daha fazladır. Bu teori, belirsiz ve doğru olmayan kavramlara ve değişkenlere matematiksel bir şekil verebilir ve güvenilmez koşullarda kararları kontrol etmek ve vermek için bir bağlam sağlar (Kosko 1992).

4.8.2.1.2. Yer Seçiminde Bulanık Mantık

Yer seçimindeki parametreler doğası gereği büyük ölçüde bulanıktır.

Bulanık mantık, yaygın olarak kullanılan yer seçimi türlerinden biridir. 0 ile 1 arasında değişen konumlara üyelik değerleri atar. 0 üyelik olmadığını veya uygun olmayan bir siteyi belirtirken, 1 üyelik veya uygun bir siteyi belirtir. Diğer site seçim yöntemleri gibi, bulanık mantık da gerekli tüm adımların takip edilmesini sağlamak için standart bir iş akışı kullanır. Diğer yöntemlerden farklıdır, çünkü çok daha karmaşıktır ve basit bir evet veya hayır yerine 0 (tamamen yanlış veya uygun değil) ve 1 (tamamen doğru veya uygun) arasında bir değerler sürekliliği kullanır. Bulanık mantık, aynı anda hem doğru hem de yanlış olabilen koşulları inceleme yeteneğine sahiptir. Bulanık mantık için standart iş akışı aşağıdaki gibidir:

1-Sorunu ve site seçim kriterlerini tanımlamak

2-Ölçüt katmanlarını toplamak

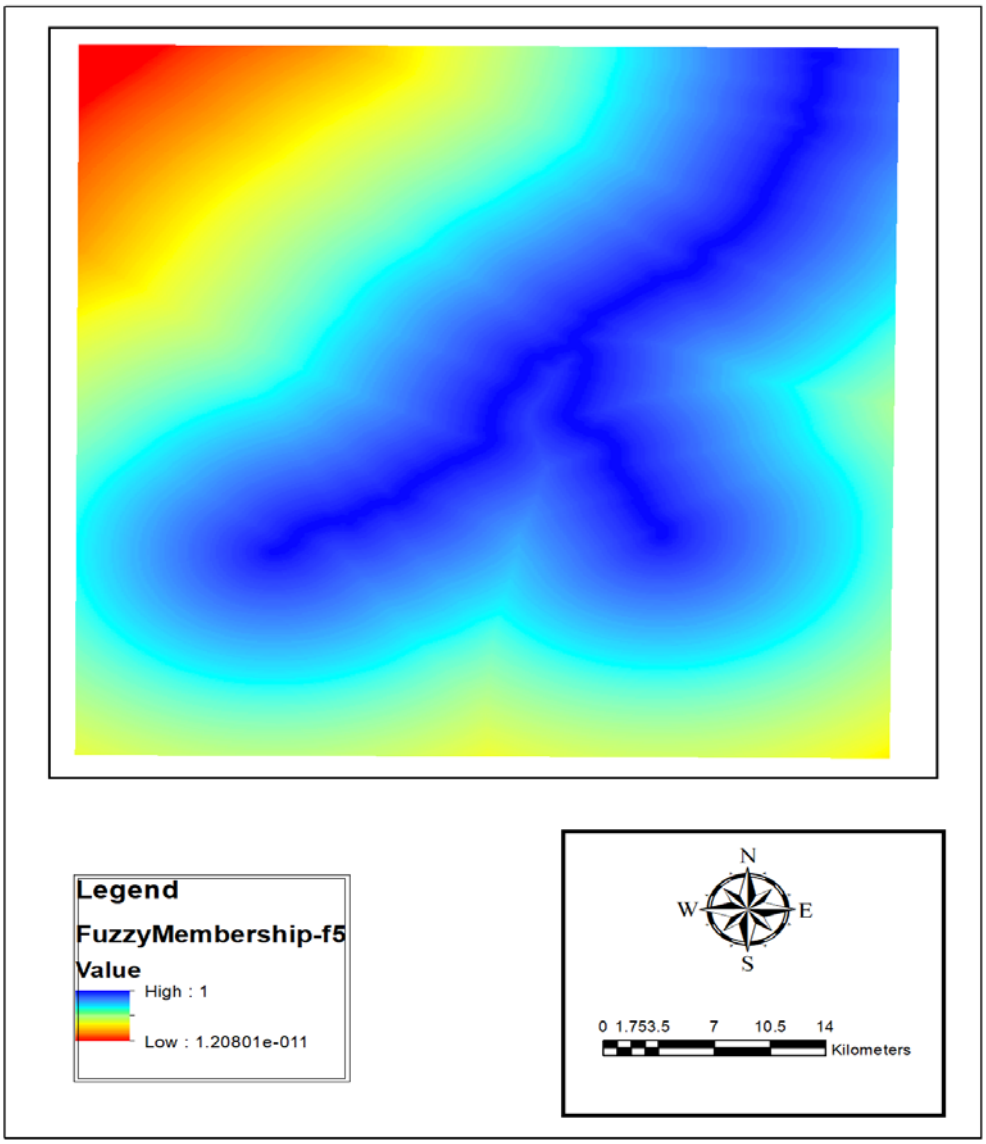
3-Bulanik üyelik değerleri atamak

4-Bulanık örtüşme yapmak

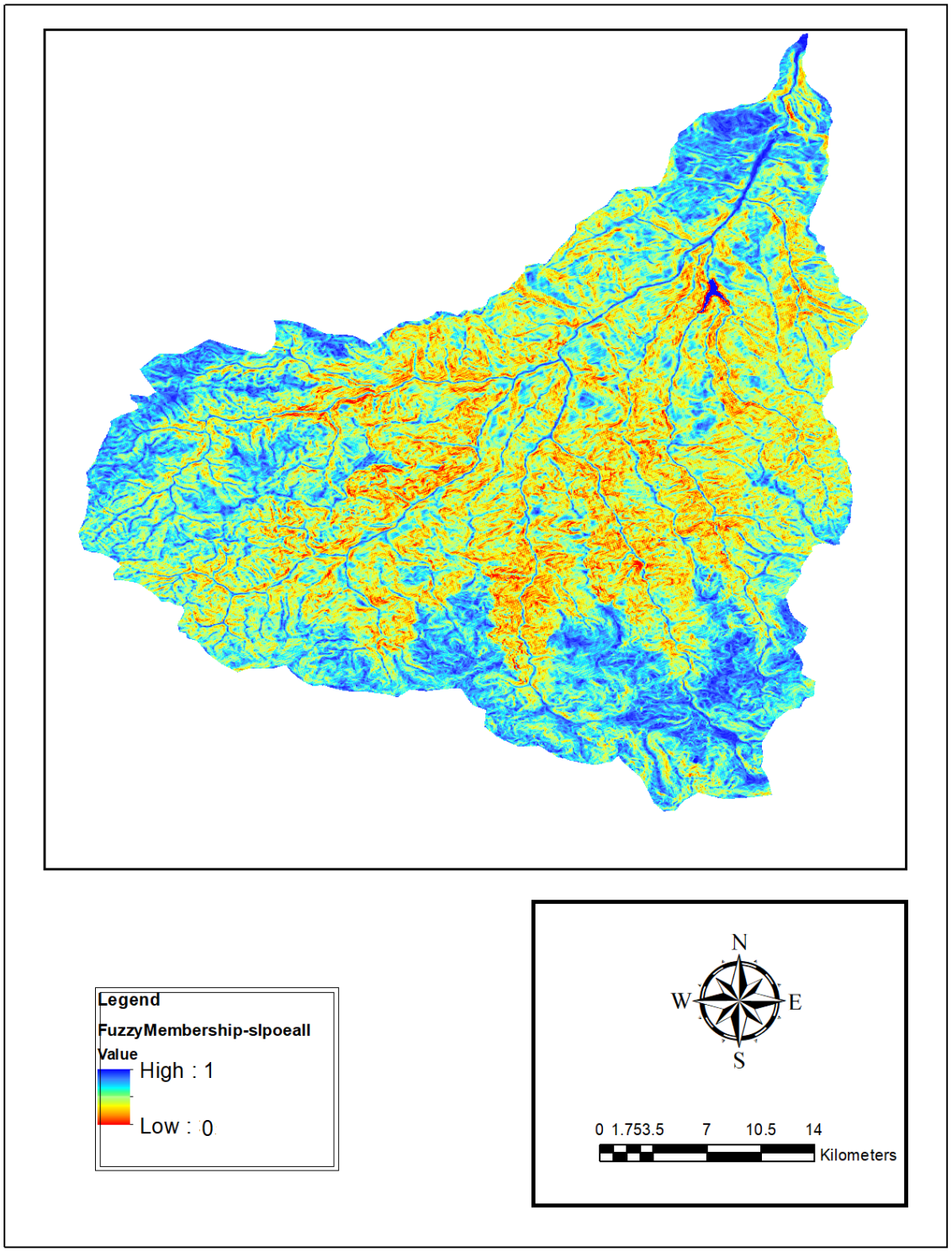
5-Sonuçları doğrulamak ve uygulamak

Bulanık mantık yöntemiyle yer seçimine başlamak için, öncelikle nehirdeki su kalitesi sensörleri için yer seçimi için etkili kriterler belirlendi. Bu kriterler şunlardır: 1- Suyu kirleten kaynaklara yakınlık 2- Su tüketen kaynaklara yakınlık 3- Çalışma alanındaki her bir alt havzanın çıkış noktasına yakınlık 4- Yollara yakınlık 5- Telefon hatlarına yakınlık 6- Eğim alan 7- Nehire Yakınlık (çünkü sensörler nehir üzerinde konumlandırılmalıdır). Bir sonraki adımda, bu kriterlerin her birinin mesafe haritası Arcgıs

yazılımında oluşturulmuş ve ardından bu mesafe haritalarının her biri, doğrusal üyelik ile bulanıktı ve sıfır ile bir arasında bir değer verildi (Doğrusal - Yüksek bulanık üyelik, büyük veya küçük değerlere atanır ve bulanık üyelik sabit bir oranda azalır Bu bulanık üyelik haritalarda en yakın yerlere 1 (en yüksek değer) ve en uzak yere 0 (en düşük değer) puan verildi. Ayrıca alanın eğim haritasında en düşük eğime, bir (en yüksek değer) puan ve 0 puan (en düşük değer) (45 derece ve üstü) verildi. Örneğin, Şekil 4.21 ve 4.22'de, yolların bulanık üyelik haritasını ve eğim bulanık üyelik haritasını gösterilmiş.

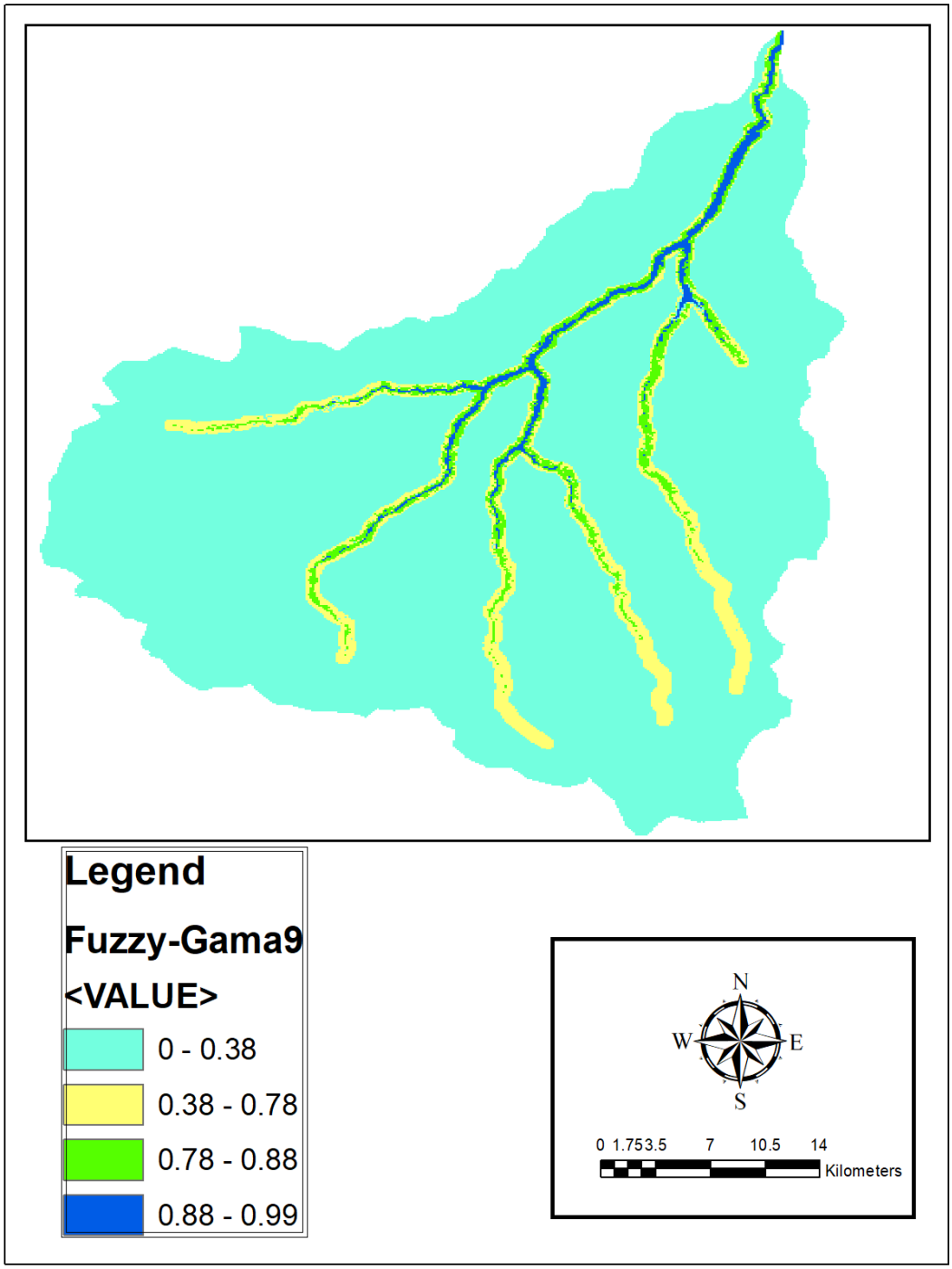


Şekil 4.21. Yolların bulanık üyelik haritası



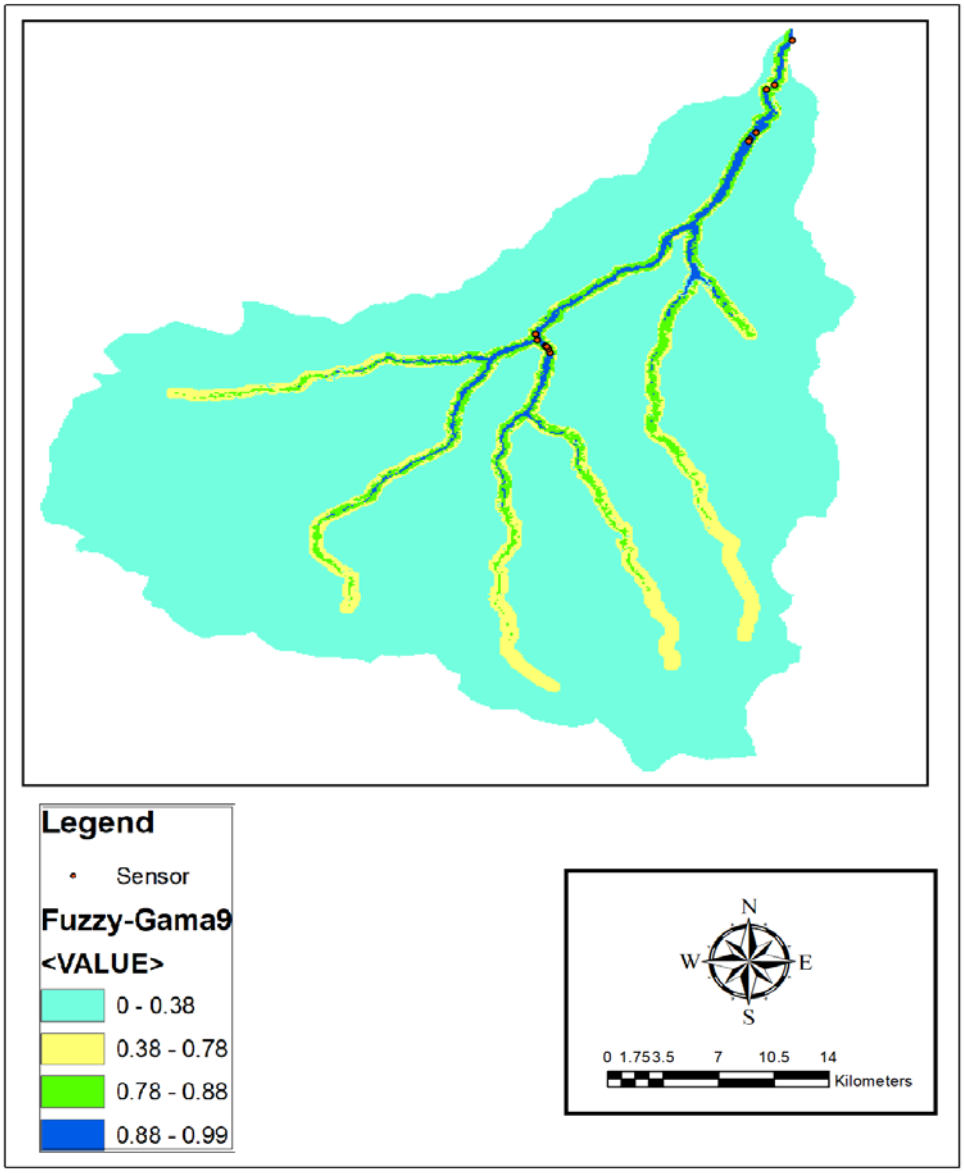
Şekil 4.22. Eğimin bulanık üyelik haritası

ve son olarak bulanık üyelik haritalari, Gamma katsayısı 0.9 ile bulanık mantık bindirme, birleştirildi ve nihai harita elde edildi.(Şekil 4.23).



Şekil 4.23. Bulanık mantık yöntemiyle su kalitesi sensörleri için uygun yerler.

Şekil 4.23'te gösterildi gibi, koyu mavi renkli yerler (0.88-0.99 değerleri) su kalitesi sensörlerini kurmak için çok uygun yerlerdir ve yeşil renkli yerler (0.78-0.88 değerinde) su kalitesi sensörleri kurmak için orta derecede uygundur ve sarı renkli yerler (0.38-0.78 değerinde) su kalitesi sensörleri kurmak için daha az tercih edilir. Şekil 4.20, NSGA-II algoritma yöntemi ve bulanık mantık sonuçlarının aynı anda gösterilmesinden elde edilen haritayı göstermektedir.



Şekil 4.24. NSGA-II algoritması ile bulanık mantık sonuçlarının karşılaştırılması

Şekil 4.24'te görebileceğiniz gibi, NSGA-II algoritma yönteminden (kırmızı noktalar) elde edilen 15 sensör lokasyonundan 14 sensör lokasyonu bulanık mantık yönteminden elde edilen çok uygun lokasyonlarda yer almaktadır (Koyu mavi ile gösterilen alanlar). Bu sonuçlar, yukarıdaki iki yöntemin% 93 uyumlu olduğunu göstermektedir.

4.9. Yapay Sinir Ağı

Bu bölümde önce uygun sinir ağının seçimi tartışılmış ve ardından GA'dan (Su kalitesi istasyonlarının uygun yerleri) elde edilen veriler sinir ağını eğitmek için kullanılmıştır ve son olarak, sinir ağının ürettiği sonuçlar, GA'nın sonuçlarıyla istatistiksel göstergelerle karşılaştırıldı.

4.9.1. Yapay Sinir Ağının Seçimi

İyi bir sinir ağı oluşturmak, belirli bir uygulama için çok önemlidir. En yaygın kullanılan yapay ağlardan biri, çok çeşitli gerçek dünya sorunlarını çözmek için kullanılan MLP sinir ağıdır. Yapay sinir ağlarının daha iyi kullanılması, içinde kullanılan parametrelerin optimizasyonunu gerektirir. Sinir ağı parametrelerinin en iyi değerlerini (katman sayısı, her katmandaki nöronlar vb.) belirlemek için, bu parametreleri deneme yanılma yoluyla kalibre etmek için çok zaman harcanır. Bu nedenle, bu çalışmada, bu parametreleri optimize etmek için, minimum hesaplama hatasıyla en uygun yöntemi ve dolayısıyla daha doğru bir cevabı elde etmek için GA kullanmıştır.

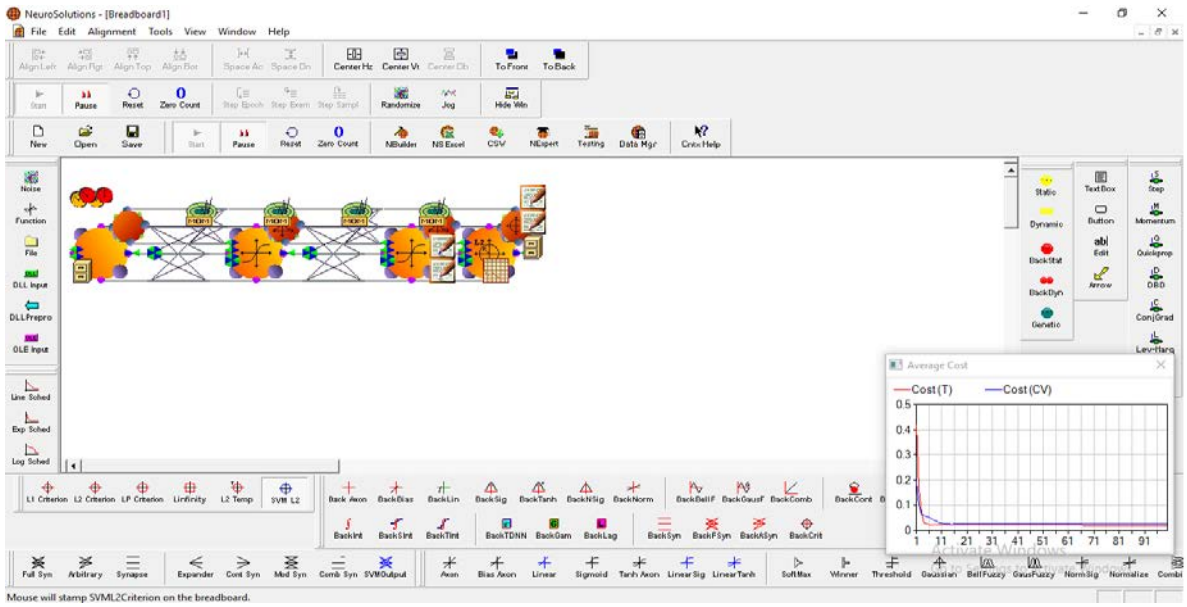
4.9.2. Kullanılmış Verilerin Hazırlaması

Bu çalışmada, GA'da kullanılan veriler, yapay sinir ağına girdi verisi olarak kullanılmıştır. Ayrıca, yapay sinir ağını eğitmek için GA'dan elde edilen su kalitesi

sensörlerinin optimum yerleri kullanıldı. Bu araştırmada ilk olarak 1- suyu kirleten kaynaklar 2- su tüketen kaynaklar 3- çalışma alanındaki her bir alt havzanın çıkış noktası 4- yollar 5- telefon hatları veya antenler verilerinden bir mesafe haritası hazirlandi. Sonra bu haritalar ASCII formatına dönüştürüldü ve ardından bu veriler normalize edildi ve son olarak yapay bir sinir ağını çalıştırmak için Excel programına girildi.

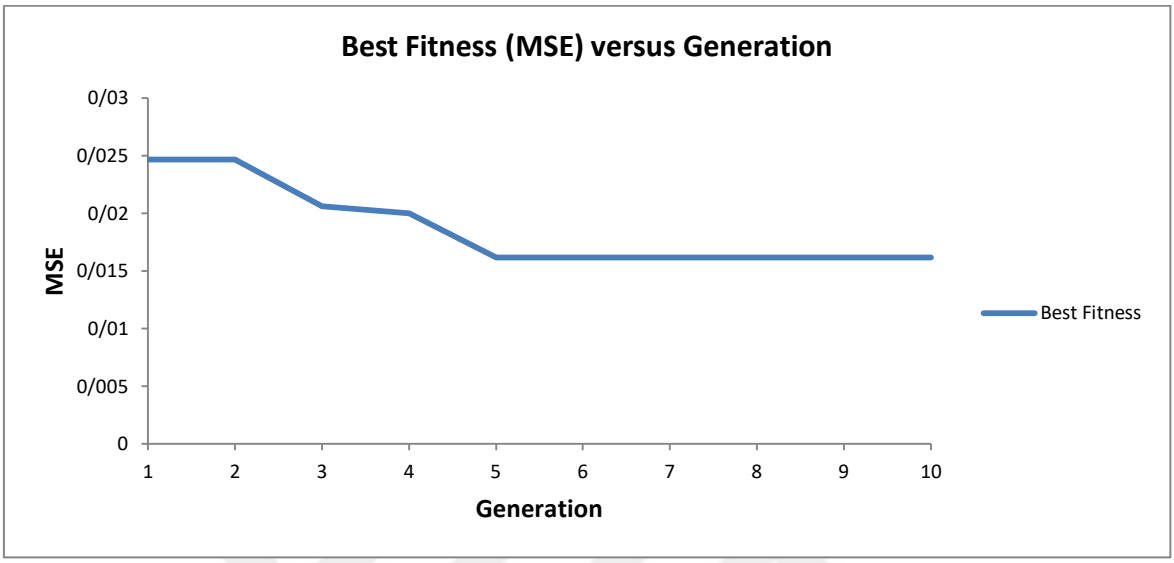
4.9.3. Yapay Sinir Ağını Çaliştirilmasi

Yapay sinir ağını çalıştırmak için önce Neuro Solutions yazılımı çalıştırıldı ve ardından Excel yazılımı ile Nero Solution arasında bağlantı kuruldu. (Şekil 4.25'te, Neuro Solutions yazılım görüntü ortamı ortamını gösterilmiş). Daha sonra bu araştırmada kullanılan veriler Excel yazılımına girilerek giriş ve çıkış verileri belirlenmiştir. Bu çalışmada, verilerin% 70'i yapay sinir ağı eğitimi,% 20'si yapay sinir ağı testi ve% 10'u cross validation için kullanıldir. Ve son olarak transfer fonksiyonları ile (Tanh, Sigmoid, linear Tanh , linear Sigmoid ve öğrenme kuralları (Momentum, Conjugate Gradient, Quickprop, Delta-Bar-Delta) çalıştırıldı. Uygulama sonuçları aşağıda verilmiştir.

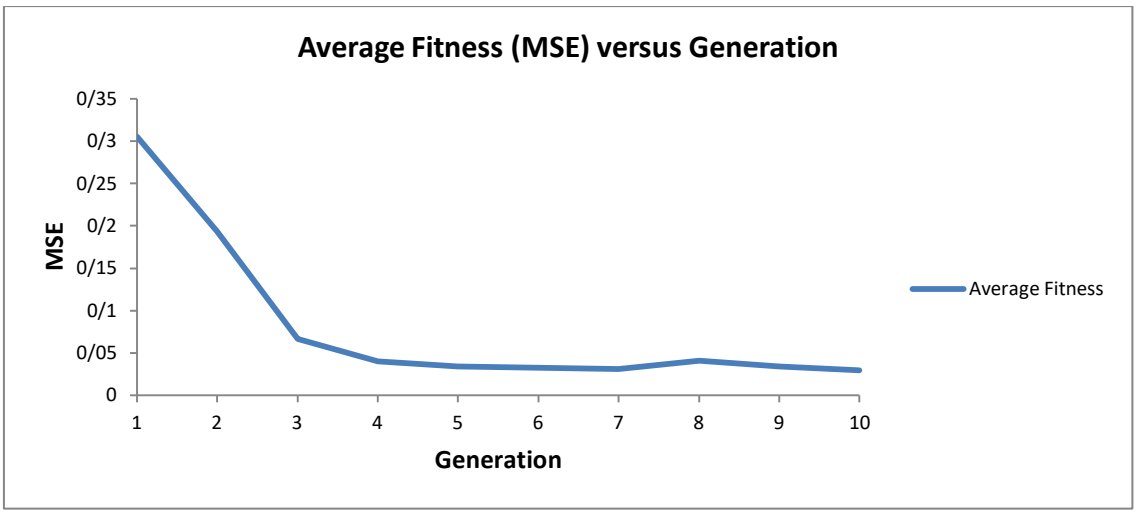


Şekil 4.25. Neuro Solutions yazılım görüntü ortamı.

4.9.3.1. Tanh Axon Transfer Fonksiyonu ve Momentum Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.26. Nesile karşı en iyi fitness (MSE)(Tanh Axon transfer fonksiyonu ve Momentum öğrenme kuralı)



Şekil 4.27. Nesile karşı ortalama fitnes (MSE) (Tanh Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

Tablo.4.4. Nesile karşı en iyi fitness ve ortalama fitnes (MSE) (Tanh Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/0246839	0/3050276
2	0/0246839	0/1931565
3	0/0205948	0/0667183
4	0/0199886	0/0403143
5	0/0161368	0/0339758
6	0/0161368	0/0323405
7	0/0161368	0/0309469
8	0/0161368	0/0405202
9	0/0161313	0/0342595
10	0/0161313	0/0295963
	Minimum Best Fitness	**Minimum Average Fitness**
	0/0161313	0/0295963

Tablo 4.5. Optimizasyon özeti(Eğitim verileri için) (Tanh Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

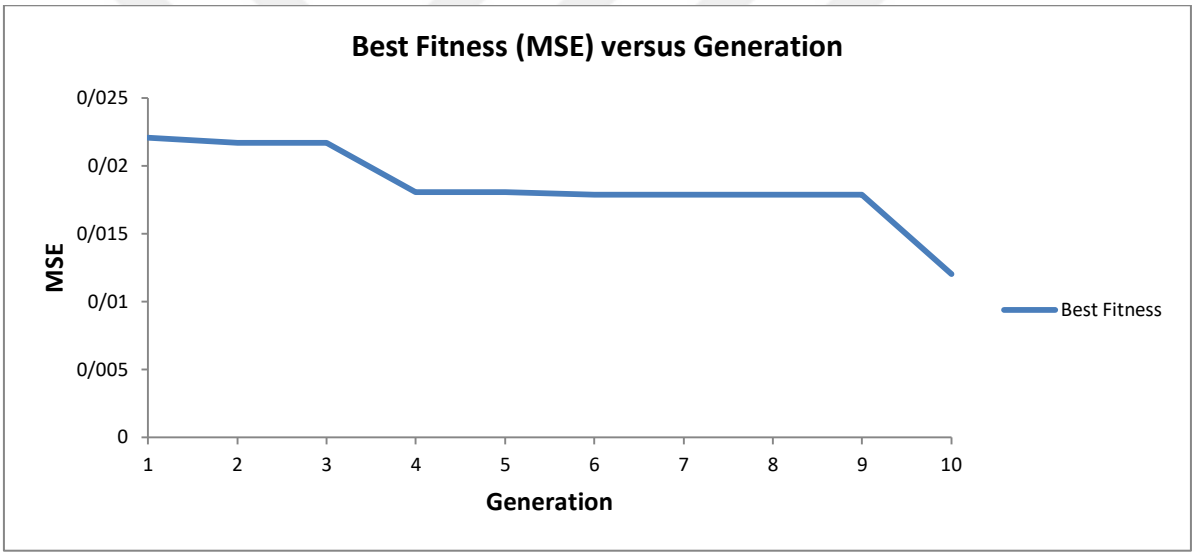
Optimization Summary	*Best Fitness*	*Average Fitness*
Generation #	9	10
Minimum MSE	0/016131394	0/029596347
Final MSE	0/016131394	0/029596347

Tablo 4.6. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için) (Tanh Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

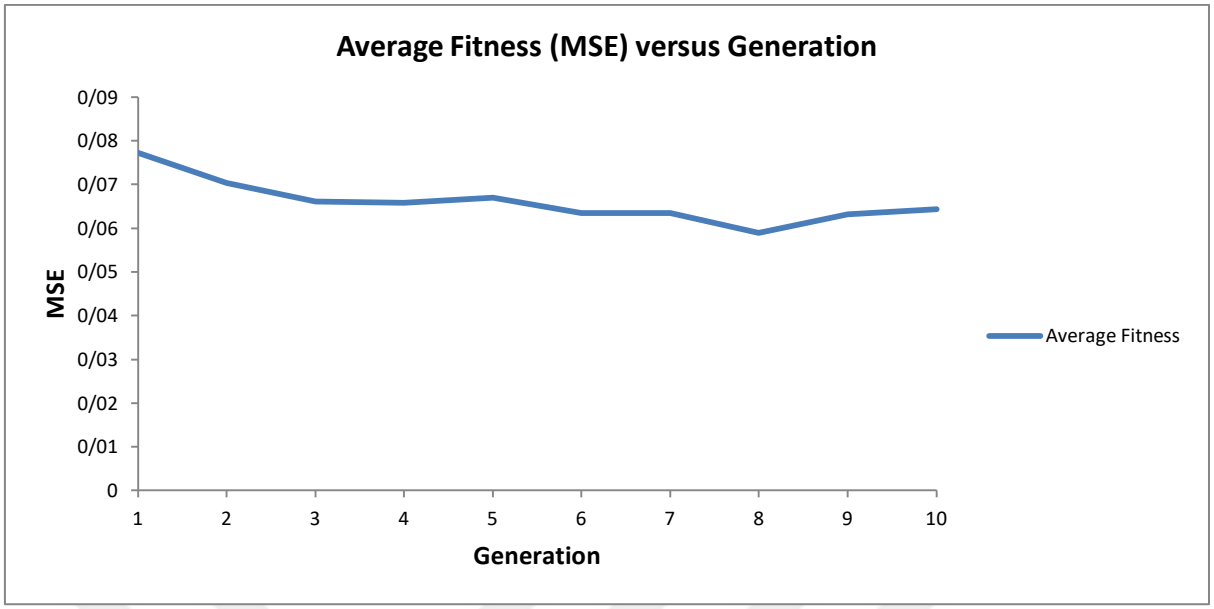
RMSE	MSE	MAE	r
0/185176	0/03429	0/17077	-0/9526

Şekil 4.26, 4.27 ve Tablo 4.4, 4.5'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı dokuzuncu nesilde ve ortalama en düşük hata oranı Onuncu nesilde eğitim verilerinde sırasıyla 0/016161394 ve 0/029596347 elde edildi.Ve test verileri için Tablo 4.6'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla -0/9526, 0/17077, 0/03429, 0/185176 elde edildi.

4.9.3.2. Sigmoid Axon Transfer Fonksiyonu ve Momentum Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.28. Nesile karşı en iyi fitness (MSE)(Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)



Şekil 4.29. Nesile karşı ortalama fitnes (MSE) (Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

Tablo 4.7. Nesile karşı en iyi fitness ve ortalama fitnes (MSE) (Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/0220929	0/0771873
2	0/0216862	0/0703006
3	0/0216862	0/0661012
4	0/0180750	0/0658466
5	0/0180750	0/0670275
6	0/0178518	0/0634737
7	0/0178518	0/0634972
8	0/0178518	0/0589784
9	0/0178518	0/0631163
10	0/0120509	0/0644178
	Minimum Best Fitness	Minimum Average Fitness
	0/0120509	0/0589784

Tablo 4.8. Optimizasyon özeti(Eğitim verileri için) (Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

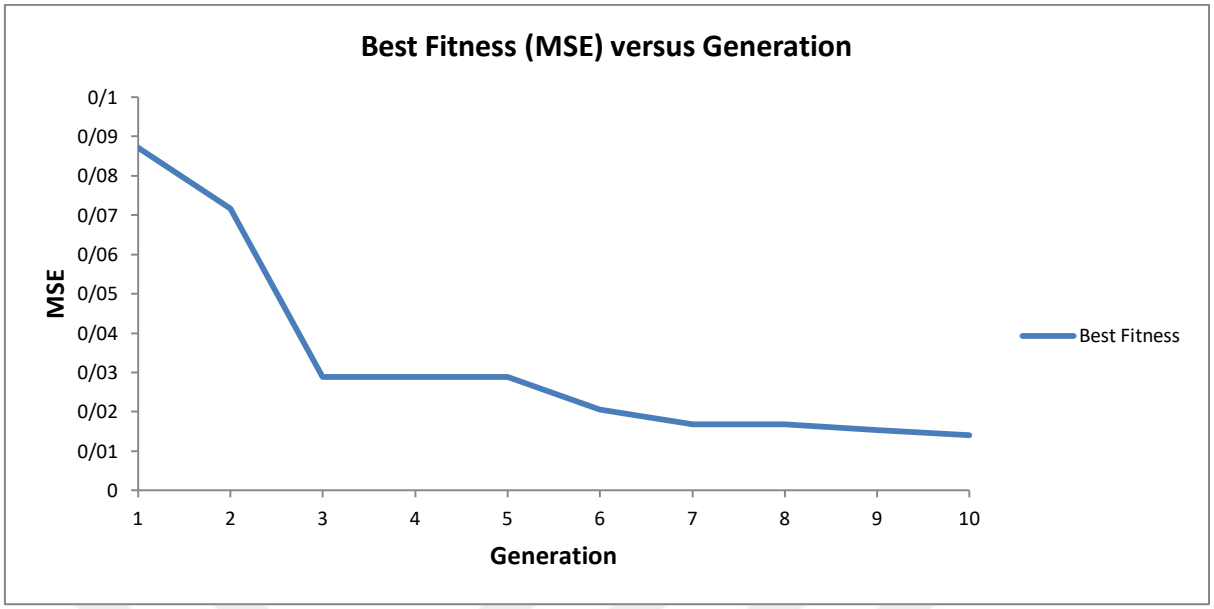
Optimization Summary	Best Fitness	Average Fitness
Generation #	10	8
Minimum MSE	0/012050905	0/058978435
Final MSE	0/012050905	0/064417824

Tablo 4.9. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için) (Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

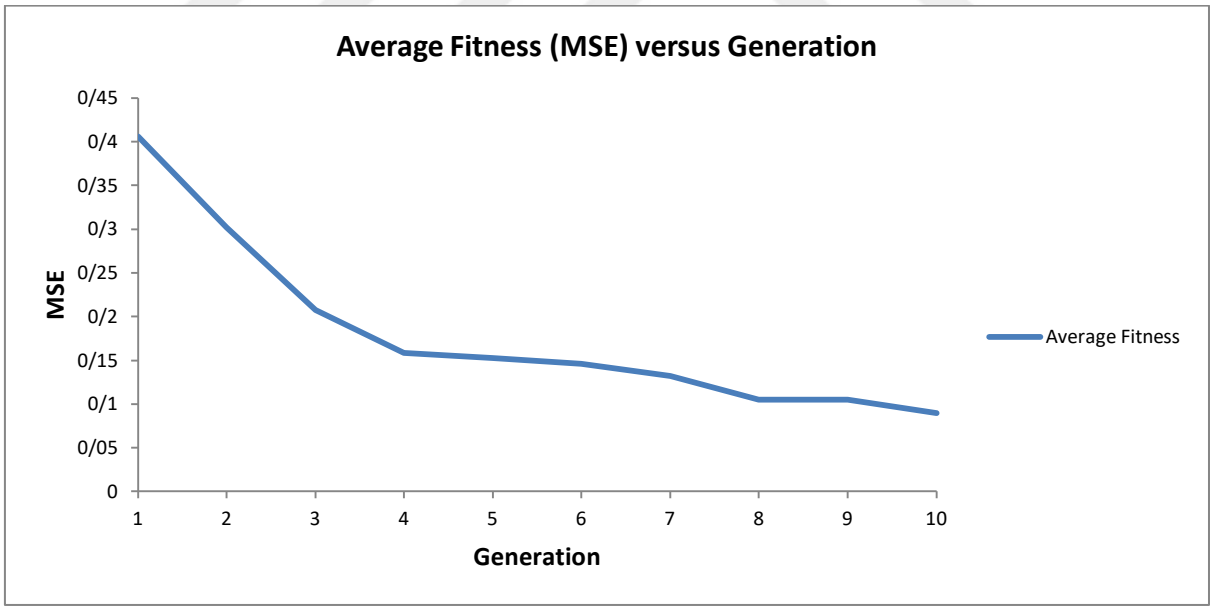
RMSE	MSE	MAE	r
0/237213	0/05627	0/22267	-0/6753

Şekil 4.28, 4.29 ve Tablo 4.7, 4.8'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı Onuncu nesilde ve ortalama en düşük hata oranı sekizinci nesilde eğitim verilerinde sırasıyla 0/01205091 ve 0/05897843 elde edildi.Ve test verileri için Tablo 4.9'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla -0/6753, 0/22267, 0/05627, 0/237213 elde edildi.

4.9.3.3. linear Tanh Axon Transfer Fonksiyonu ve Momentum Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.30. Nesile karşı en iyi fitness (MSE)(linear Tanh axon transfer fonksiyonu ve Momentum öğrenme kuralı)



Şekil 4.31. Nesile karşı ortalama fitness (MSE)(linear Tanh axon transfer fonksiyonu ve Momentum öğrenme kuralı)

Tablo 4.10. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(linear Tanh axon transfer fonksiyonu ve Momentum öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/0871086	0/4060871
2	0/07170437	0/30164981
3	0/02889104	0/20742105
4	0/02889104	0/15817516
5	0/02889104	0/15229969
6	0/02057495	0/14614739
7	0/01689284	0/13230415
8	0/01689284	0/10482556
9	0/01542682	0/10483905
10	0/01406243	0/0897278
	Minimum Best Fitness	Minimum Average Fitness
	0/01406243	0/0897278

Tablo 4.11. Optimizasyon özeti(Eğitim verileri için)(linear Tanh axon transfer fonksiyonu ve Momentum öğrenme kuralı)

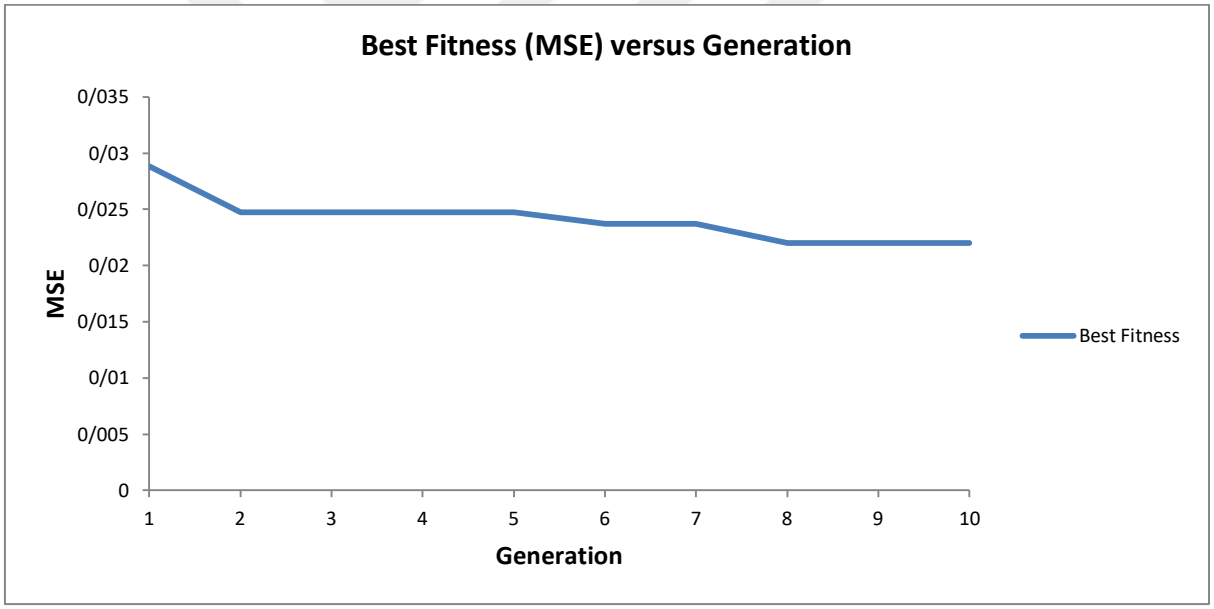
Optimization Summary	*Best Fitness*	*Average Fitness*
Generation #	10	10
Minimum MSE	0/014062425	0/089727797
Final MSE	0/014062425	0/089727797

Tablo 4.12. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(linear Tanh axon transfer fonksiyonu ve Momentum öğrenme kuralı)

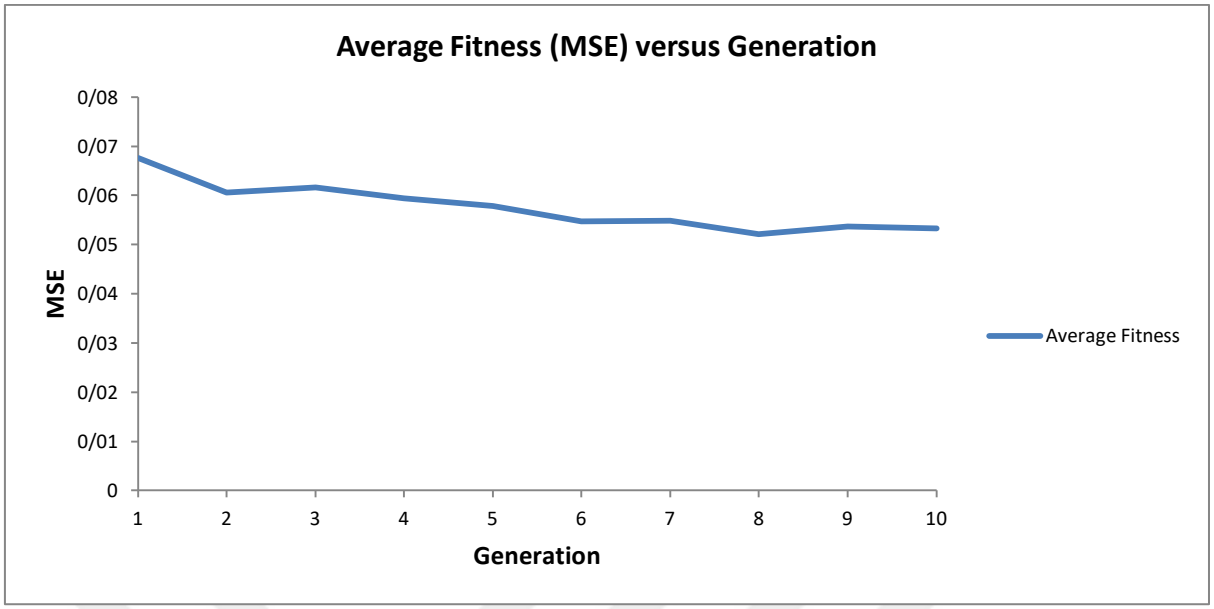
RMSE	MSE	MAE	r
0/213659	0/04565	0/19888	0/05686

Şekil 4.30, 4.31 ve Tablo 4.10, 4.11'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı Onuncu nesilde ve ortalama en düşük hata oranı Onuncu nesilde eğitim verilerinde sırasıyla 0/01406243 ve 0/0897278 elde edildi.Ve test verileri için Tablo 4.12'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0/05686, 0/19888, 0/04565, 0/213659 elde edildi.

4.9.3.4. linear Sigmoid Axon Transfer Fonksiyonu ve Momentum Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.32. Nesile karşı en iyi fitness (MSE)(linear Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)



Şekil 4.33. Nesile karşı ortalama fitness (MSE)(linear Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

Tablo 4.13. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(linear Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/02881676	0/0676469
2	0/02475846	0/06057893
3	0/02475846	0/06164122
4	0/02475846	0/05936819
5	0/02475846	0/05782387
6	0/02370873	0/0546854
7	0/02370873	0/05491774
8	0/02200273	0/05214954
9	0/02200273	0/05374764
10	0/02200273	0/05323666

Minimum Best Fitness	Minimum Average Fitness
0/02200273	0/05214954

Tablo 4.14. Optimizasyon özeti(Eğitim verileri için)(linear Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

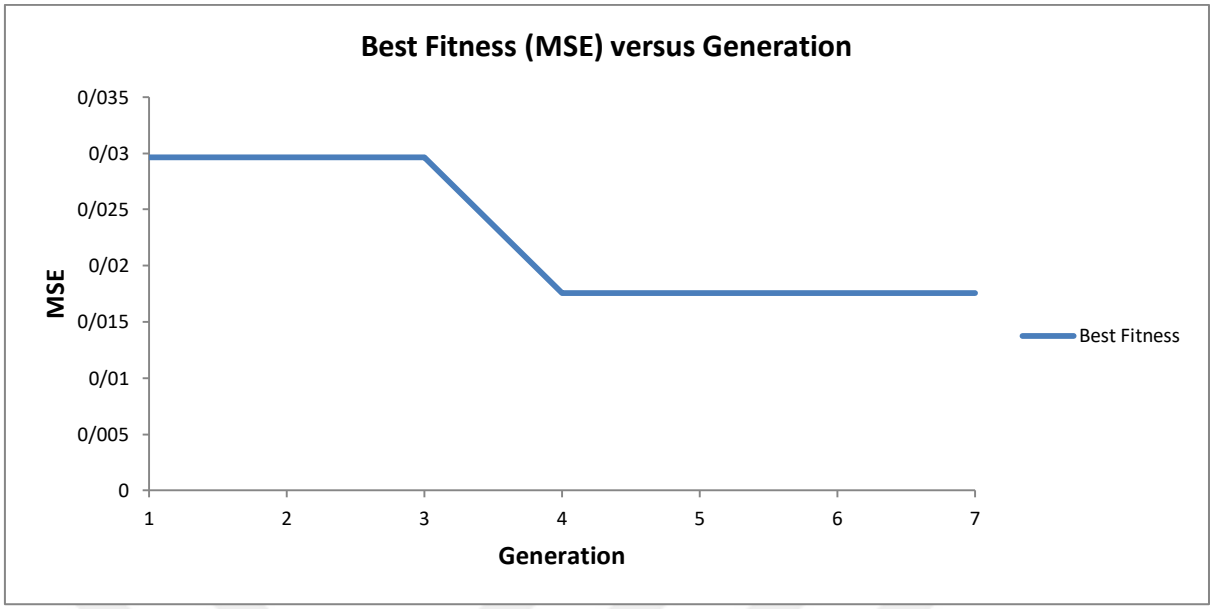
Optimization Summary	Best Fitness	Average Fitness
Generation #	8	8
Minimum MSE	0/022002727	0/052149542
Final MSE	0/022002727	0/053236656

Tablo 4.15. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(linear Sigmoid Axon transfer fonksiyonu ve Momentum öğrenme kuralı)

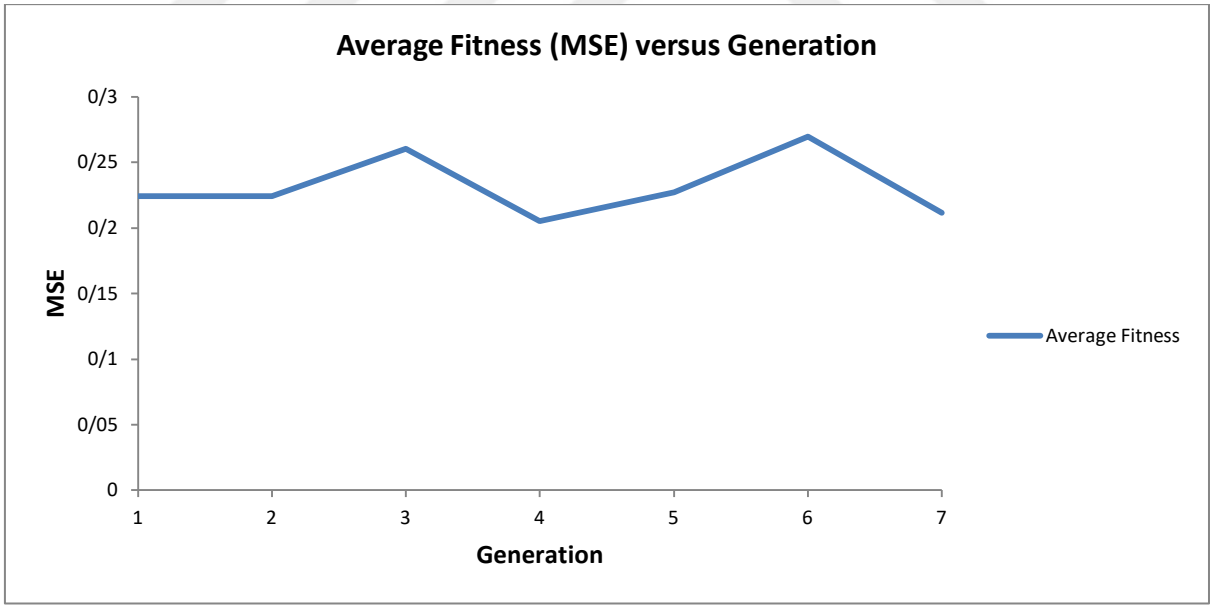
RMSE	MSE	MAE	r
0/143736	0/02066	0/12597	0/22167

Şekil 4.32, 4.33 ve Tablo 4.13, 4.14'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı sekizinci nesilde ve ortalama en düşük hata oranı sekizinci nesilde eğitim verilerinde sırasıyla 0/02200273 ve 0/05214954 elde edildi.Ve test verileri için Tablo 4.15'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0/22167, 0/12597, 0/02066, 0/143736 elde edildi.

4.9.3.5. Tanh Axon Transfer Fonksiyonu ve Conjugate Gradient Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.34. Nesile karşı en iyi fitness (MSE)(Tanh Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)



Şekil 4.35. Nesile karşı ortalama fitness (MSE)(Tanh Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

Tablo 4.16. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(Tanh Axon transfer fonksiyonu ve Conjugate Gradient kuralı) öğrenme

Nesil	Best Fitness	Average Fitness
1	0/02962588	0/22419657
2	0/02962588	0/22427401
3	0/02962588	0/26053327
4	0/01756763	0/20499477
5	0/01756763	0/22712824
6	0/01756763	0/26962456
7	0/01756763	0/21153513
	Minimum Best Fitness	**Minimum Average Fitness**
	0/01756763	0/20499477

Tablo 4.17. Optimizasyon özeti(Eğitim verileri için)(Tanh Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

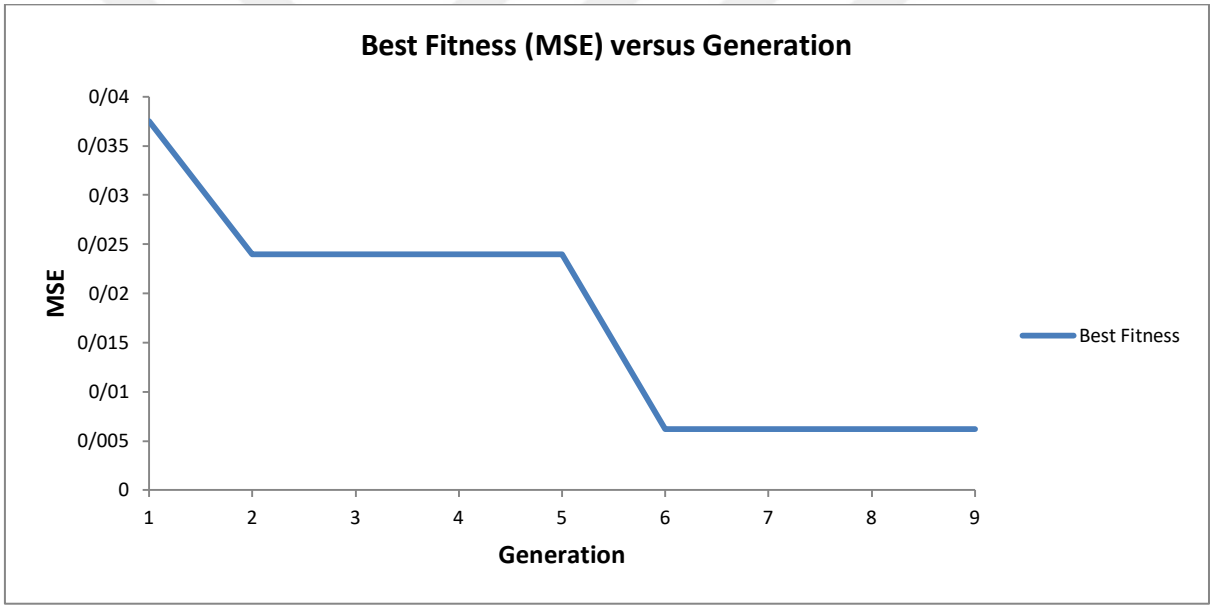
Optimization Summary	*Best Fitness*	*Average Fitness*
Generation #	4	4
Minimum MSE	0/017567629	0/204994768
Final MSE	0/017567629	0/211535126

Tablo 4.18. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(Tanh Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

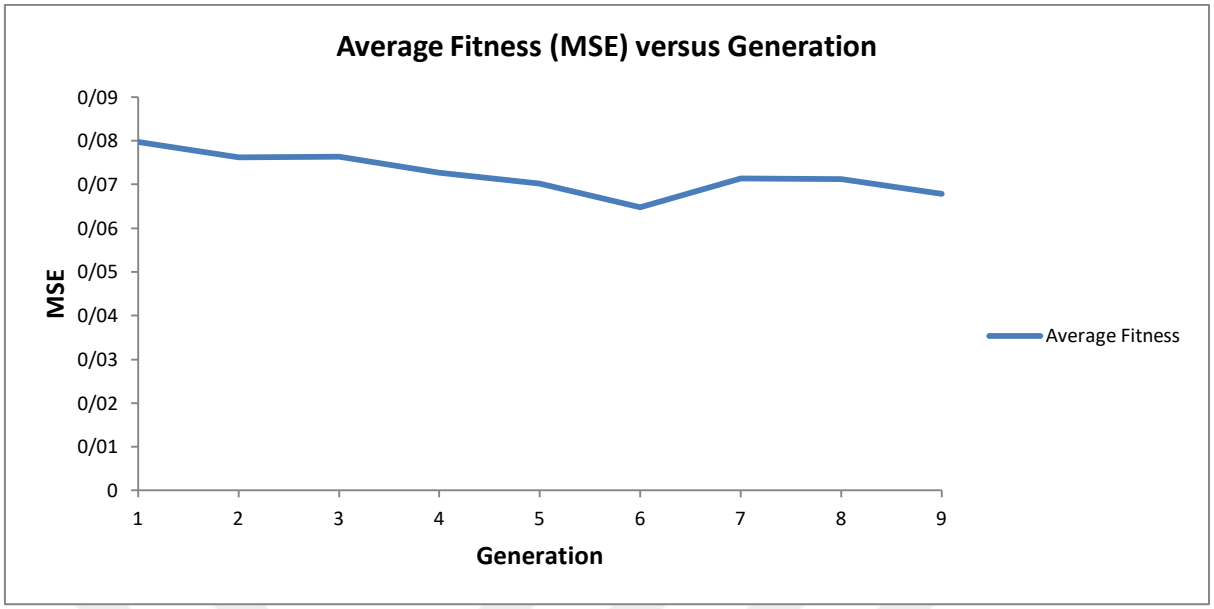
RMSE	MSE	MAE	r
0/161895	0/02621	0/14512	-0/12682

Şekil 4.34, 4.35 ve Tablo 4.16, 4.17'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı dördüncü nesilde ve ortalama en düşük hata oranı dördüncü nesilde eğitim verilerinde sırasıyla 0/01756763 ve 0/20499477 elde edildi.Ve test verileri için Tablo 4.18'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla -0/12682, 0/14512, 0/02621, 0/161895 elde edildi.

4.9.3.6. Sigmoid Axon Transfer Fonksiyonu ve Conjugate Gradient Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.36. Nesile karşı en iyi fitness (MSE)(Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)



Şekil 4.37. Nesile karşı ortalama fitness (MSE)(Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

Tablo 4.19. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/03751228	0/07977846
2	0/02398081	0/07627284
3	0/02398081	0/07631105
4	0/02398081	0/07269005
5	0/02398081	0/07026006
6	0/00620552	0/06478619
7	0/00620552	0/07134764
8	0/00620552	0/07123562
9	0/00620552	0/06779973

Minimum Best Fitness	Minimum Average Fitness
0/00620552	0/06478619

Tablo 4.20. Optimizasyon özeti(Eğitim verileri için)(Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

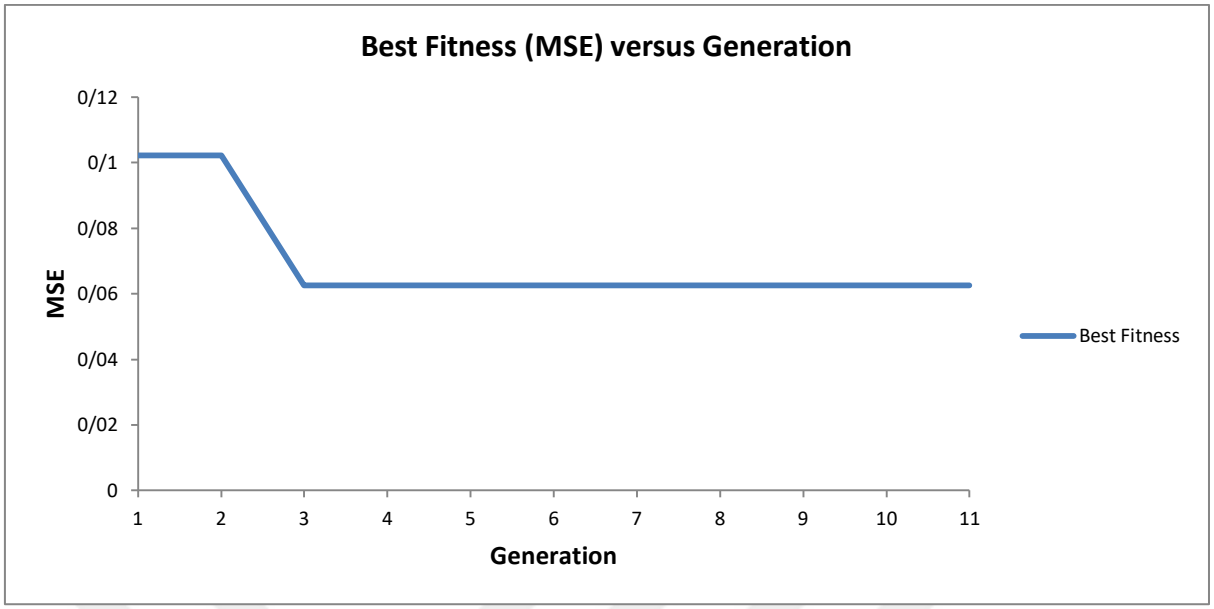
Optimization Summary	Best Fitness	Average Fitness
Generation #	6	6
Minimum MSE	0/00620552	0/064786188
Final MSE	0/00620552	0/067799732

Tablo 4.21. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

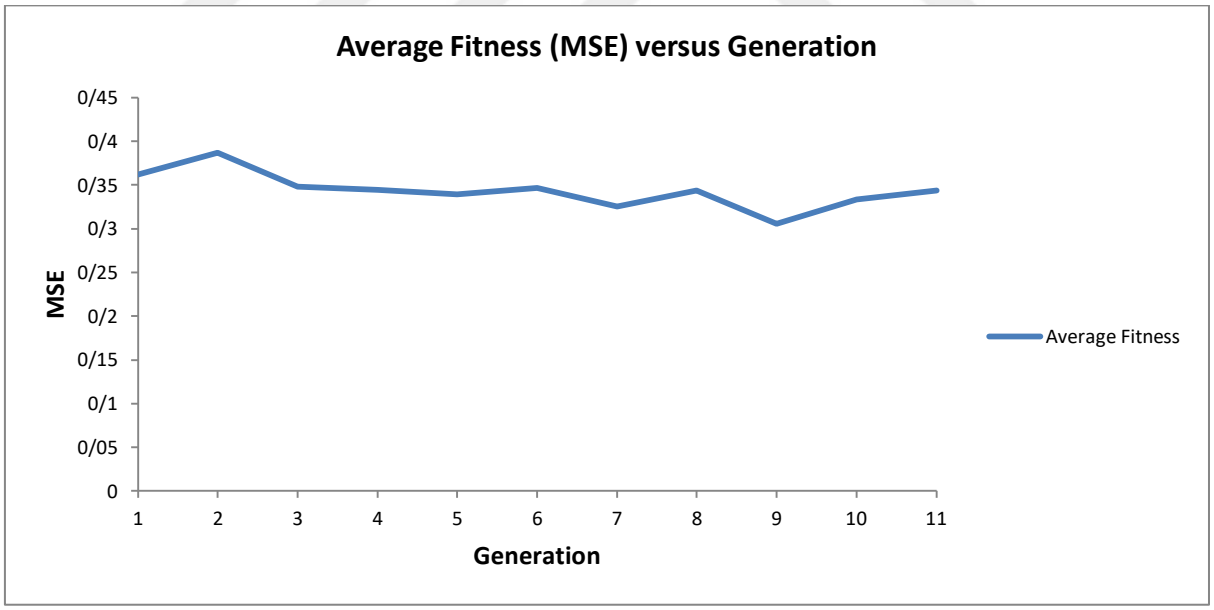
RMSE	MSE	MAE	r
0/163432	0/02671	0/1452	0/73705

Şekil 4.36, 4.37 ve Tablo 4.19, 4.20'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı altıncı nesilde ve ortalama en düşük hata oranı altıncı nesilde eğitim verilerinde sırasıyla 0/00620552 ve 0/06478619 elde edildi.Ve test verileri için Tablo 4.21'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0/73705, 0/1452, 0/02671, 0/163432 elde edildi.

4.9.3.7. linear Tanh Axon Transfer Fonksiyonu ve Conjugate Gradient Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.38. Nesile karşı en iyi fitness (MSE)(linear Tanh axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)



Şekil 4.39. Nesile karşı ortalama fitness (MSE)(linear Tanh axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

Tablo 4.22. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(linear Tanh axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/10223604	0/36230993
2	0/10223604	0/38686523
3	0/06251613	0/34824255
4	0/06251613	0/34430206
5	0/06251613	0/33961874
6	0/06251613	0/3466841
7	0/06251613	0/32514685
8	0/06251613	0/34367579
9	0/06251613	0/30595347
10	0/06251613	0/33324862
11	0/06251613	0/34362203
	Minimum Best Fitness	**Minimum Average Fitness**
	0/06251613	0/30595347

Tablo 4.23. Optimizasyon özeti(Eğitim verileri için)(linear Tanh axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

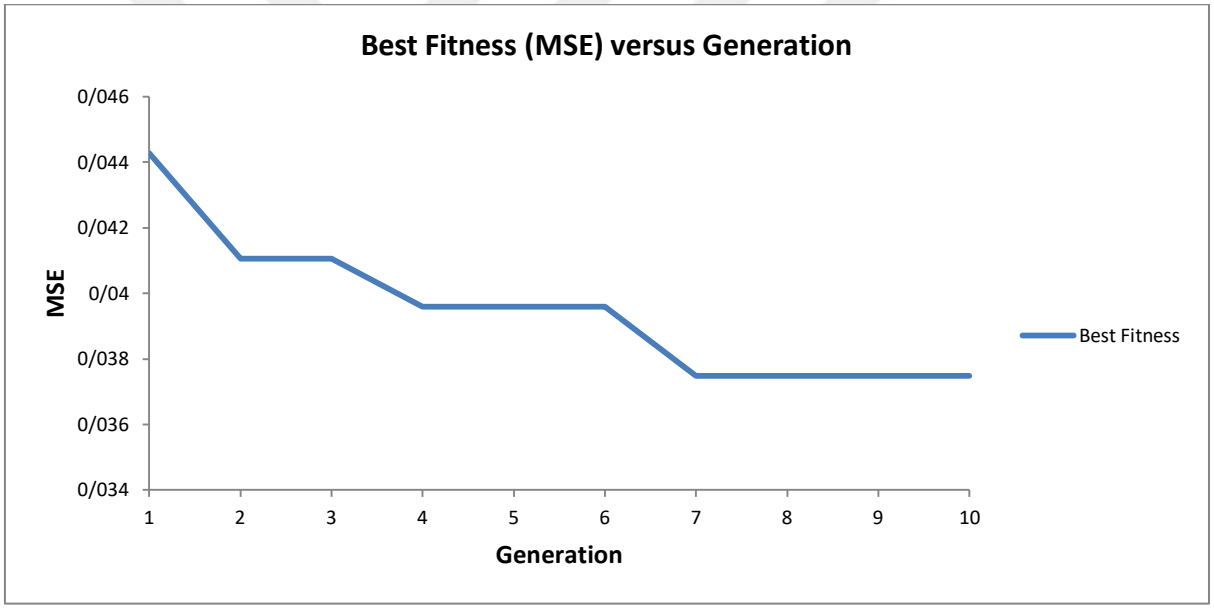
Optimization Summary	*Best Fitness*	*Average Fitness*
Generation #	3	9
Minimum MSE	0/062516131	0/305953473
Final MSE	0/062516131	0/343622029

Tablo 4.24. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(linear Tanh axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

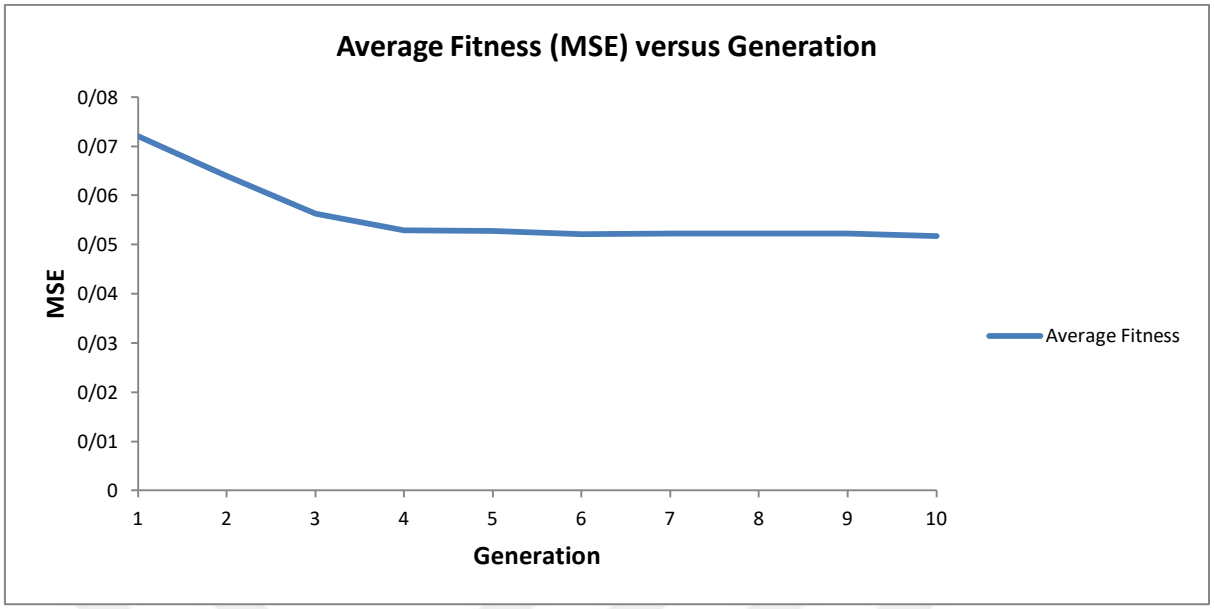
RMSE	MSE	MAE	r
0/21192	0/04491	0/18292	0/28192

Şekil 4.38, 4.39 ve Tablo 4.22, 4.23'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı üçüncü nesilde ve ortalama en düşük hata oranı dokuzuncu nesilde eğitim verilerinde sırasıyla 0/06251613 ve 0/30595347elde edildi.Ve test verileri için Tablo 4.24'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0/28192, 0/1452, 0/04491, 0/28192 elde edildi.

4.9.3.8. linear Sigmoid Axon Transfer Fonksiyonu ve Conjugate Gradient Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.40. Nesile karşı en iyi fitness (MSE)(linear Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)



Şekil 4.41. Nesile karşı ortalama fitness (MSE)(linear Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

Tablo 4.25. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(linear Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/04427511	0/07202516
2	0/04105735	0/06403229
3	0/04105735	0/05626028
4	0/039593	0/05293036
5	0/039593	0/05282433
6	0/039593	0/05213122
7	0/03748564	0/05221761
8	0/03748564	0/05221824
9	0/03748564	0/05228779
10	0/03748564	0/05168293

Minimum Best Fitness	Minimum Average Fitness
0/03748564	0/05168293

Tablo 4.26. Optimizasyon özeti(Eğitim verileri için)(linear Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

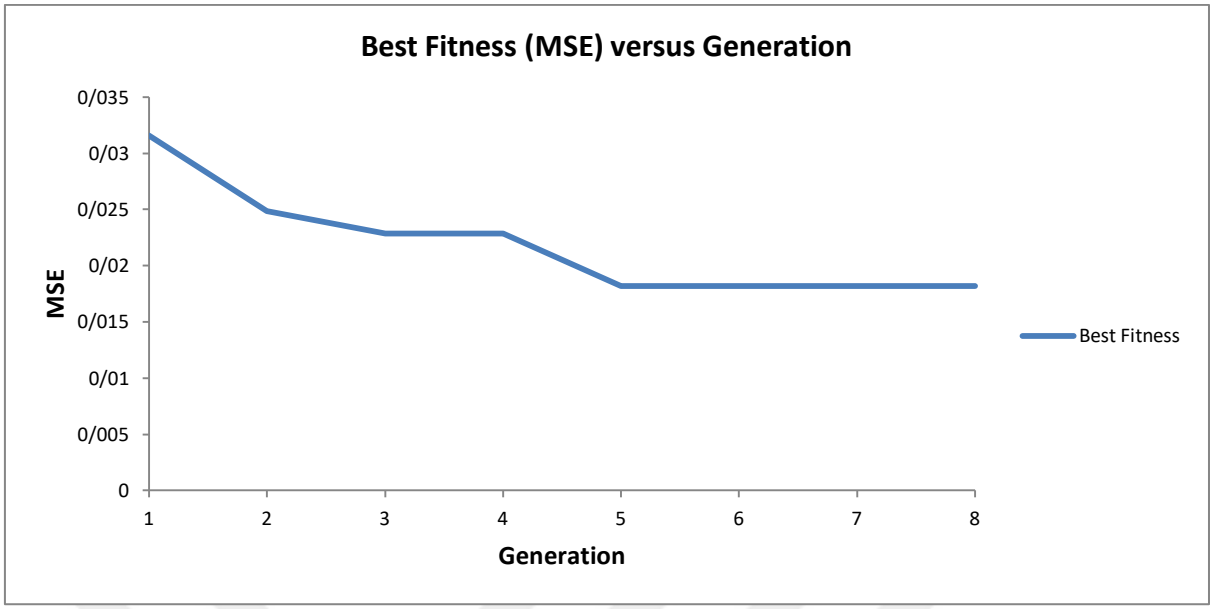
Optimization Summary	Best Fitness	Average Fitness
Generation #	7	10
Minimum MSE	0/037485644	0/05168293
Final MSE	0/037485644	0/05168293

Tablo 4.27. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(linear Sigmoid Axon transfer fonksiyonu ve Conjugate Gradient öğrenme kuralı)

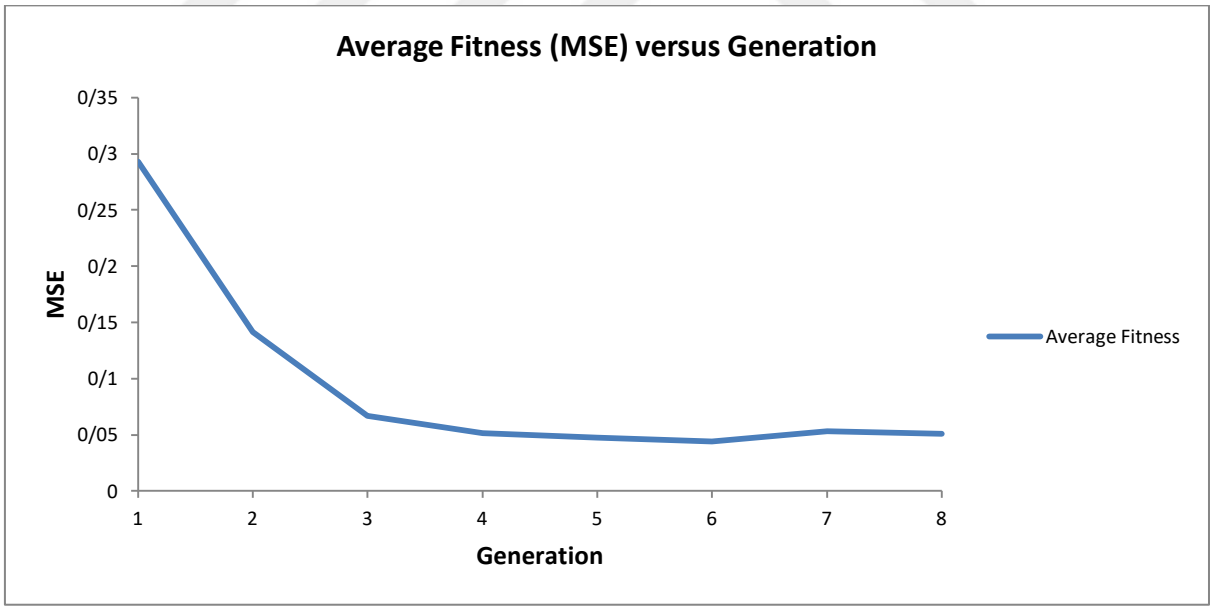
RMSE	MSE	MAE	r
0/142021	0/02017	0/12453	0/22023

Şekil 4.40, 4.41 ve Tablo 4.25, 4.26'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı yedinci nesilde ve ortalama en düşük hata oranı onuncu nesilde eğitim verilerinde sırasıyla 0/03748564 ve 0/05168293 elde edildi.Ve test verileri için Tablo 4.27'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0/22023, 0/12453, 0/02017, 0/142021 elde edildi.

4.9.3.9. Tanh Axon Transfer Fonksiyonu ve Quickprop Öğrenme kuralı ile Çok Katmanlı Perceptronı



Şekil 4.42. Nesile karşı en iyi fitness (MSE)(Tanh Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)



Şekil 4.43. Nesile karşı ortalama fitness (MSE)(Tanh Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

Tablo 4.28. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(Tanh Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/03159342	0/2927725
2	0/02483211	0/14146988
3	0/02288216	0/06665131
4	0/02288216	0/05129079
5	0/01816615	0/04740701
6	0/01816615	0/04421751
7	0/01816615	0/05287824
8	0/01816615	0/05106459
	Minimum Best Fitness	Minimum Average Fitness
	0/01816615	0/04421751

Tablo 4.29. Optimizasyon özeti(Eğitim verileri için)(Tanh Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

Optimization Summary	*Best Fitness*	*Average Fitness*
Generation #	5	6
Minimum MSE	0/018166153	0/044217512
Final MSE	0/018166153	0/051064588

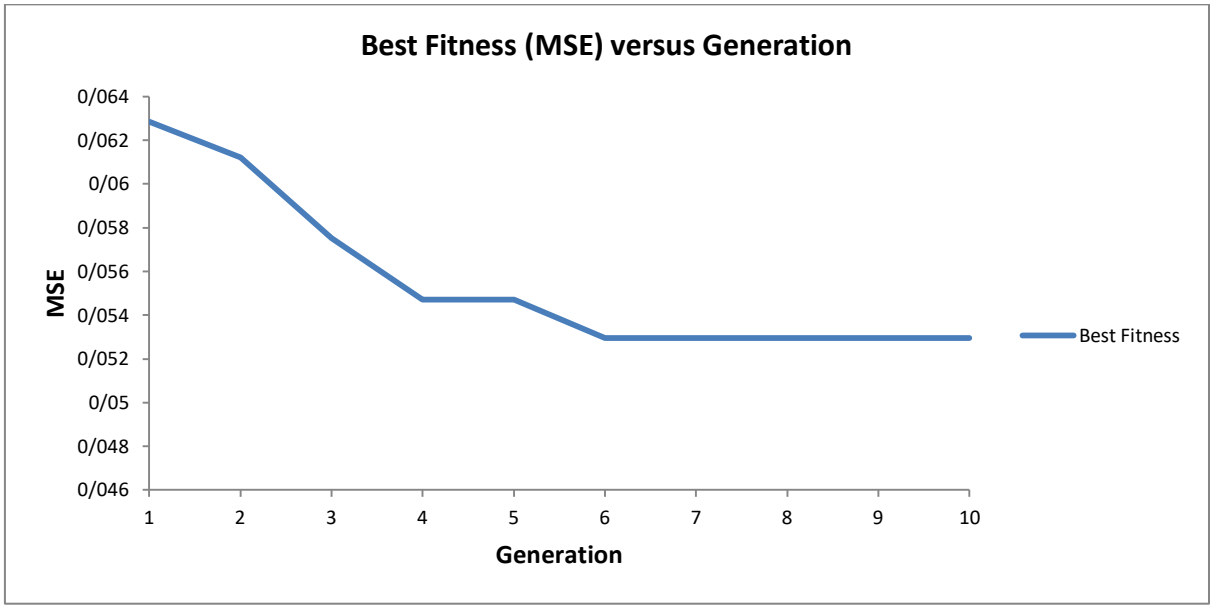
Tablo 4.30. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(Tanh Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

RMSE	MSE	MAE	r
0/16562	0/02743	0/14721	-0/31479

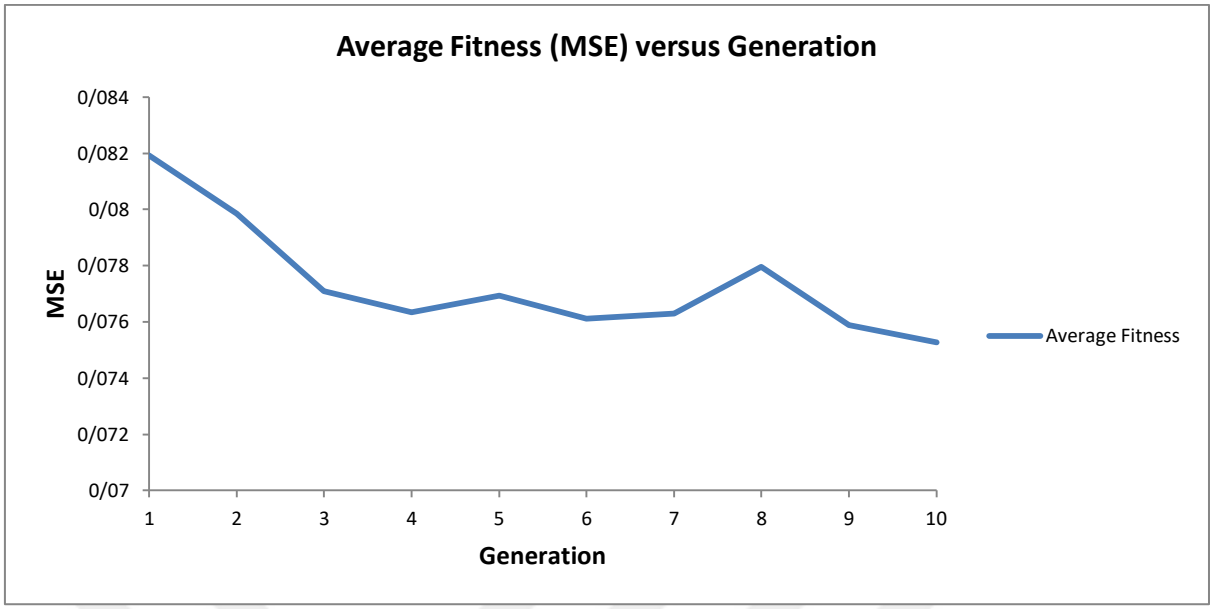
Şekil 4.42, 4.43 ve Tablo 4.28, 4.29'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı beşinci nesilde ve ortalama en düşük hata oranı altıncı nesilde eğitim

verilerinde sırasıyla 0/01816615 ve 0/04421751 elde edildi.Ve test verileri için Tablo 4.30'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla -0/31479, 0/14721, 0/02743, 0/16562 elde edildi.

4.9.3.10. Sigmoid Axon Transfer Fonksiyonu ve Quickprop Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.44. Nesile karşı en iyi fitness (MSE)(Sigmoid Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)



Şekil 4.45. Nesile karşı ortalama fitness (MSE)(Sigmoid Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

Tablo 4.31. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(Sigmoid Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/06284486	0/08191872
2	0/06121205	0/07984196
3	0/0575331	0/07708564
4	0/0547201	0/07634821
5	0/0547201	0/07692283
6	0/05293841	0/07611783
7	0/05293841	0/07629516
8	0/05293841	0/07796428
9	0/05293841	0/07588669
10	0/05293841	0/07527712
	Minimum Best Fitness	Minimum Average Fitness
	0/05293841	0/07527712

Tablo 4.32. Optimizasyon özeti(Eğitim verileri için)(Sigmoid Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

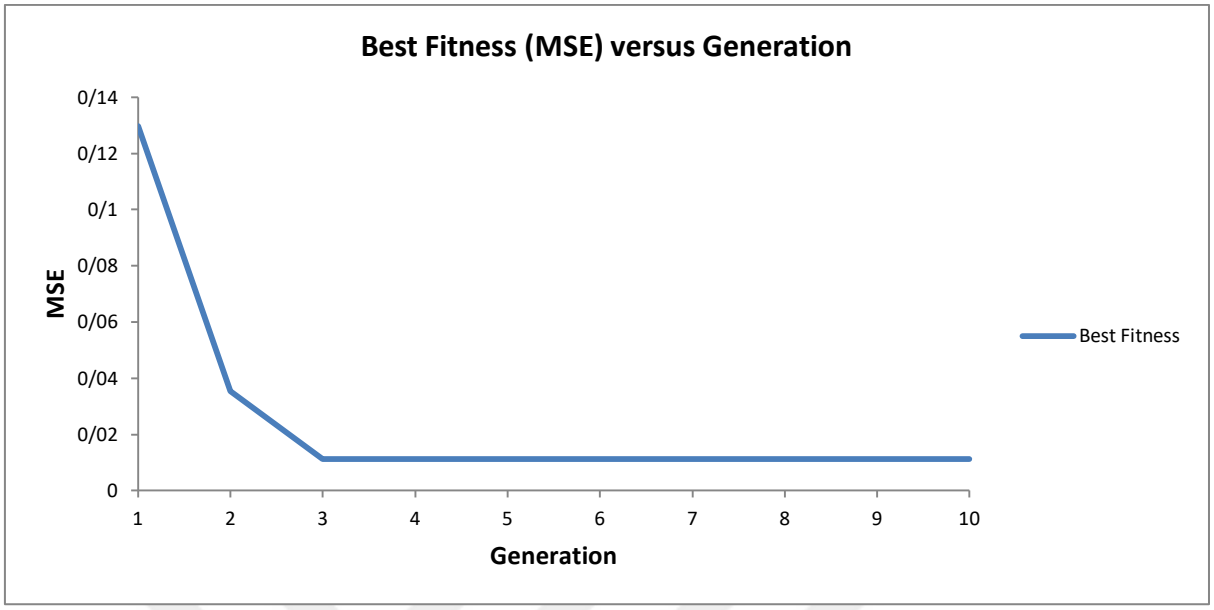
Optimization Summary	Best Fitness	Average Fitness
Generation #	6	10
Minimum MSE	0/052938409	0/07527712
Final MSE	0/052938409	0/07527712

Tablo 4.33. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(Sigmoid Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

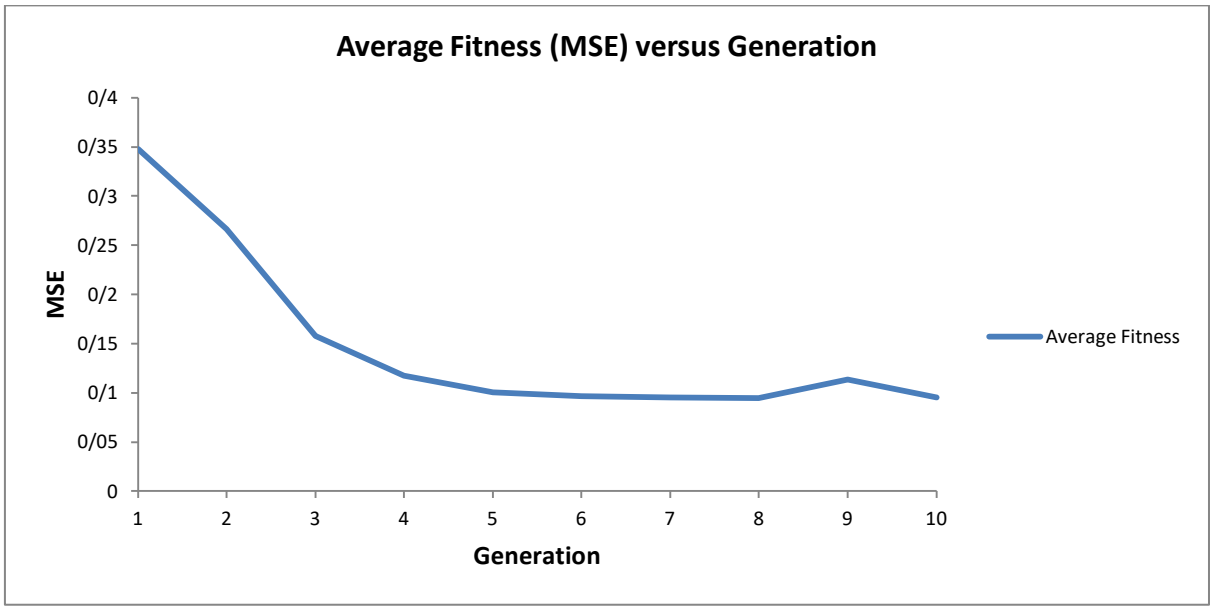
RMSE	MSE	MAE	r
0/301496	0/0909	0/29094	-0/86952

Şekil 4.44, 4.45 ve Tablo 4.31, 4.32'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı altıncı nesilde ve ortalama en düşük hata oranı onuncu nesilde eğitim verilerinde sırasıyla 0/05293841 ve 0/07527712 elde edildi.Ve test verileri için Tablo 4.33'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla -0/86952, 0/29094, 0/0909, 0/301496 elde edildi.

4.9.3.11. linear Tanh Axon Transfer Fonksiyonu ve Quickprop Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.46. Nesile karşı en iyi fitness (MSE)(linear Tanh axon transfer fonksiyonu ve Quickprop öğrenme kuralı)



Şekil 4.47. Nesile karşı ortalama fitness (MSE)(linear Tanh axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

Tablo 4.34. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(linear Tanh axon transfer fonksiyonu ve Quickprop kuralı öğrenme)

Nesil	Best Fitness	Average Fitness
1	0/129803	0/34788013
2	0/03538634	0/26673448
3	0/01132006	0/15747823
4	0/01132006	0/11742253
5	0/01132006	0/10039497
6	0/01132006	0/09659334
7	0/01132006	0/09552099
8	0/01132006	0/09466372
9	0/01132006	0/11348162
10	0/01132006	0/09526879
	Minimum Best Fitness	**Minimum Average Fitness**
	0/01132006	0/09466372

Tablo 4.35. Optimizasyon özeti(Eğitim verileri için)(linear Tanh axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

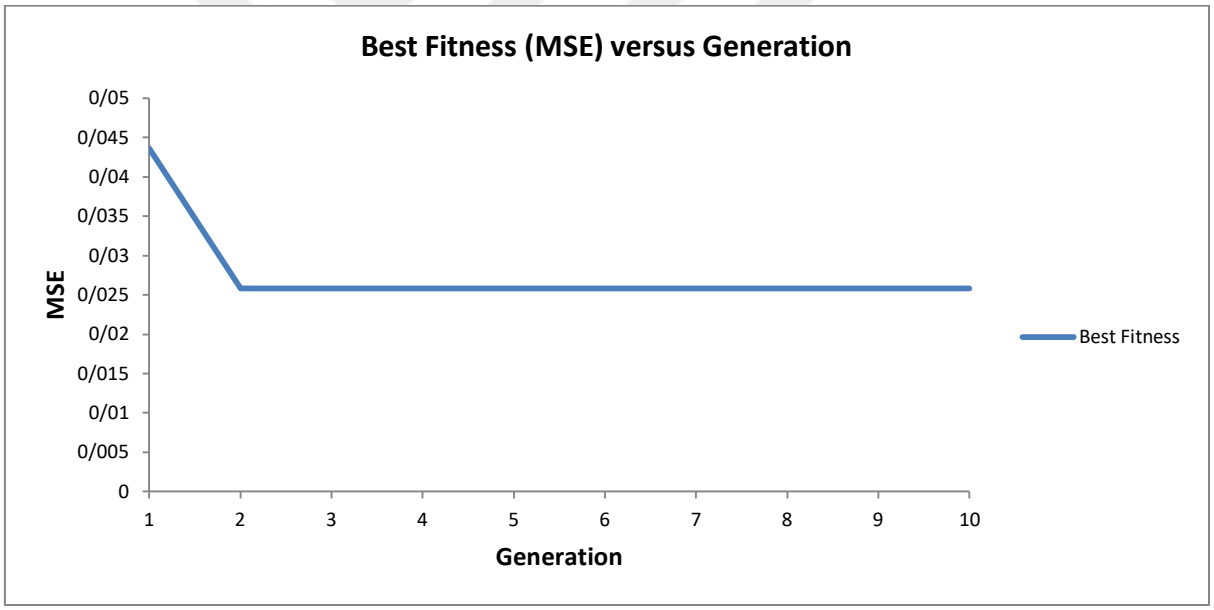
Optimization Summary	*Best Fitness*	*Average Fitness*
Generation #	3	8
Minimum MSE	0/01132006	0/094663724
Final MSE	0/01132006	0/095268793

Tablo 4.36. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(linear Tanh axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

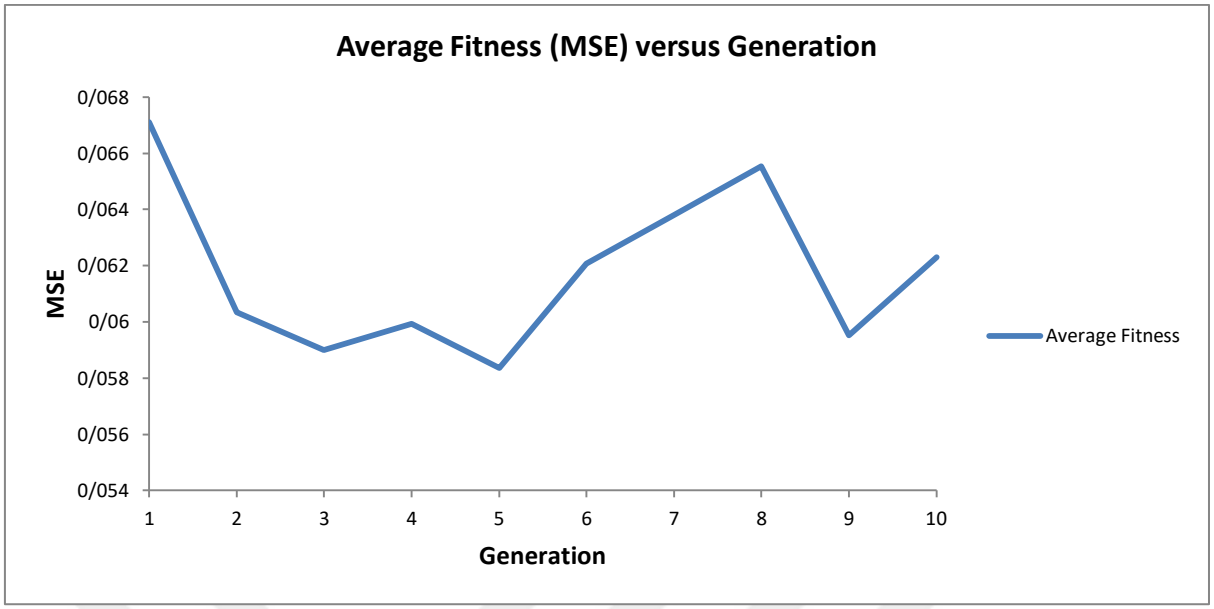
RMSE	MSE	MAE	r
0/218266	0/04764	0/19692	0/288

Şekil 4.46, 4.47 ve Tablo 4.34, 4.35'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı üçüncü nesilde ve ortalama en düşük hata oranı sekizinci nesilde eğitim verilerinde sırasıyla 0/01132006 ve 0/09466372 elde edildi.Ve test verileri için Tablo 4.36'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0/288, 0/19692, 0/04764, 0/218266 elde edildi.

4.9.3.12. linear Sigmoid Axon Transfer Fonksiyonu ve Quickprop Ğğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.48. Nesile karşı en iyi fitness (MSE)(linear Sigmoid Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)



Şekil 4.49. Nesile karşı ortalama fitnes (MSE)(linear Sigmoid Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

Tablo 4.37. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(linear Sigmoid Axon transfer fonksiyonu ve kuralı) Quickprop öğrenme

Nesil	Best Fitness	Average Fitness
1	0/04365038	0/06710171
2	0/0258224	0/06035199
3	0/0258224	0/05899071
4	0/0258224	0/05994188
5	0/0258224	0/05836878
6	0/0258224	0/06208296
7	0/0258224	0/06379663
8	0/0258224	0/06554583
9	0/0258224	0/05952181
10	0/0258224	0/0622905
	Minimum Best Fitness	Minimum Average Fitness
	0/0258224	0/05836878

Tablo 4.38. Optimizasyon özeti(Eğitim verileri için)(linear Sigmoid Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

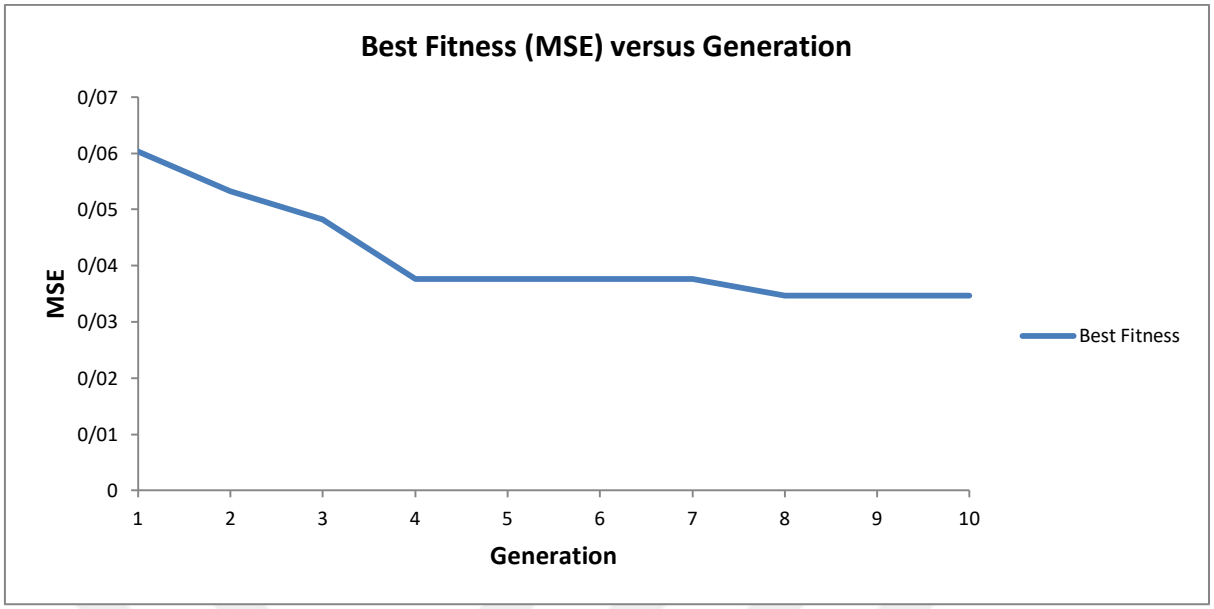
Optimization Summary	Best Fitness	Average Fitness
Generation #	2	5
Minimum MSE	0/025822399	0/058368776
Final MSE	0/025822399	0/062290497

4.39. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(linear Sigmoid Axon transfer fonksiyonu ve Quickprop öğrenme kuralı)

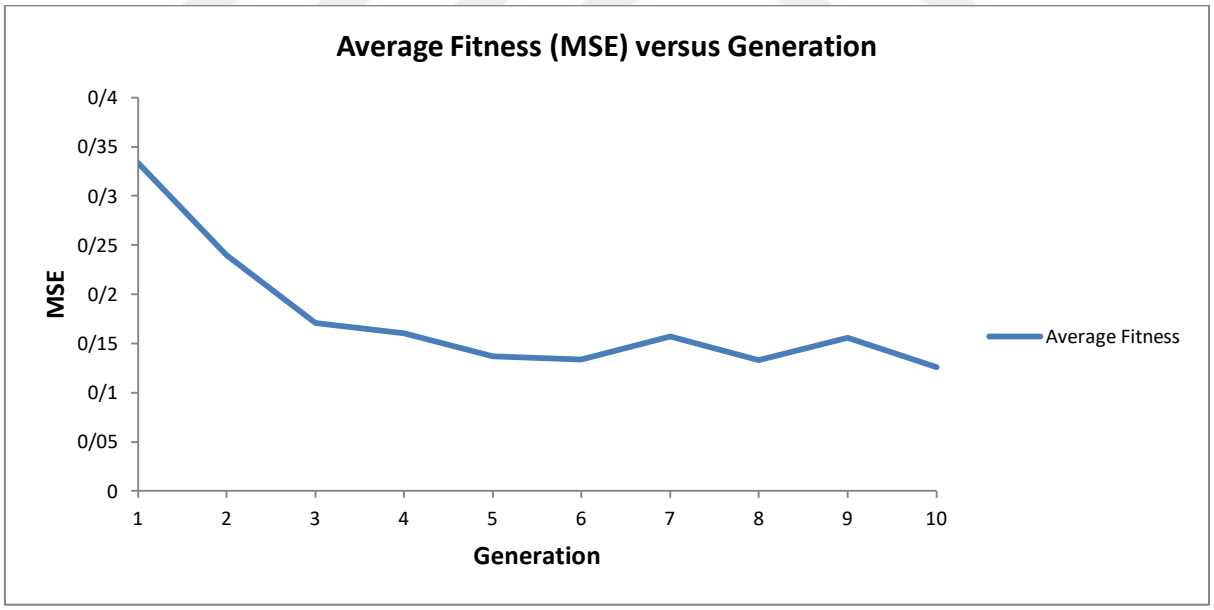
RMSE	MSE	MAE	r
0/454511	0/20658	0/12597	0/22167

Şekil 4.48, 4.49 ve Tablo 4.37, 4.38'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı İkinci nesilde ve ortalama en düşük hata oranı beşinci nesilde eğitim verilerinde sırasıyla 0/0258224 ve 0/05836878 elde edildi.Ve test verileri için Tablo 4.39'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0/22167, 0/12597, 0/20658,

4.9.3.13. Tanh Axon Transfer Fonksiyonu Delta-Bar-Delta Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.50. Nesile karşı en iyi fitness (MSE)(Tanh Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)



Şekil 4.51. Nesile karşı ortalama fitness (MSE)(Tanh Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

Tablo 4.40. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(Tanh Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/06029745	0/33371326
2	0/05326168	0/23951535
3	0/04827561	0/17089422
4	0/03768751	0/16022025
5	0/03768751	0/13668454
6	0/03768751	0/13390161
7	0/03768751	0/15732759
8	0/03464483	0/13294056
9	0/03464483	0/15558182
10	0/03464483	0/12557493
	Minimum Best Fitness	Minimum Average Fitness
	0/03464483	0/12557493

Tablo 4.41. Optimizasyon özeti(Eğitim verileri için)(Tanh Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

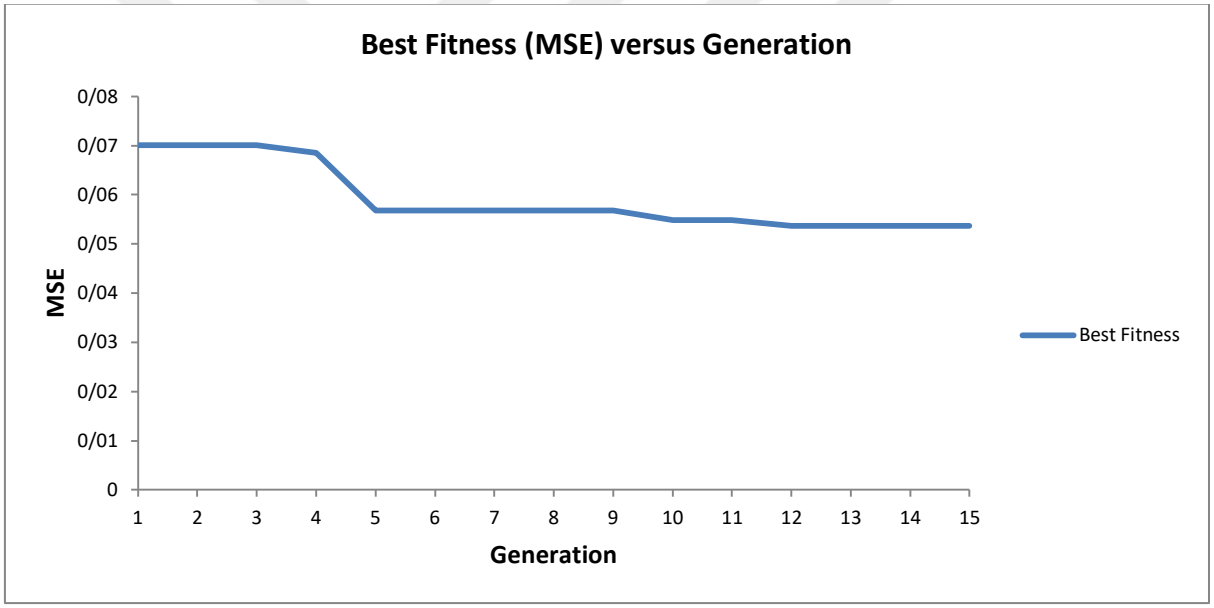
Optimization Summary	*Best Fitness*	*Average Fitness*
Generation #	8	10
Minimum MSE	0/034644827	0/125574932
Final MSE	0/034644827	0/125574932

Tablo 4.42. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(Tanh Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

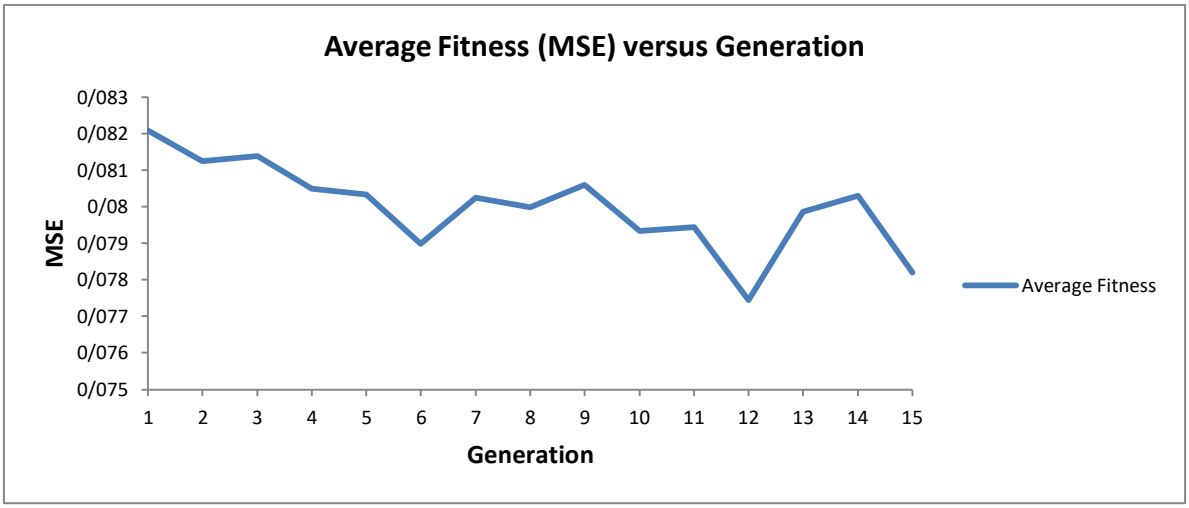
RMSE	MSE	MAE	r
0/819311	0/67127	0/2257	-0/56396

Şekil 4.50, 4.51 ve Tablo 4.40, 4.41'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı sekizinci nesilde ve ortalama en düşük hata oranı onuncu nesilde eğitim verilerinde sırasıyla 0/03464483 ve 0/12557493 elde edildi.Ve test verileri için Tablo 4.42'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla -0/56396, 0/2257, 0/67127, 0/819311 elde edildi.

4.9.3.14. Sigmoid Axon Transfer Fonksiyonu Delta-Bar-Delta Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.52. Nesile karşı en iyi fitness (MSE)(Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)



Şekil 4.53. Nesile karşı ortalama fitness (MSE)(Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

Tablo 4.43. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/07009359	0/08208717
2	0/07009359	0/08124978
3	0/07009359	0/08138549
4	0/06858876	0/08049323
5	0/05686615	0/08032775
6	0/05686615	0/07899113
7	0/05686615	0/08024998
8	0/05686615	0/07998185
9	0/05686615	0/08060228
10	0/05488976	0/07932819
11	0/05488976	0/07944056
12	0/05374157	0/07744838
13	0/05374157	0/07986845
14	0/05374157	0/08029658
15	0/05374157	0/07819727
	Minimum Best Fitness	**Minimum Average Fitness**
	0/05374157	0/07744838

Tablo 4.44. Optimizasyon özeti(Eğitim verileri için)(Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

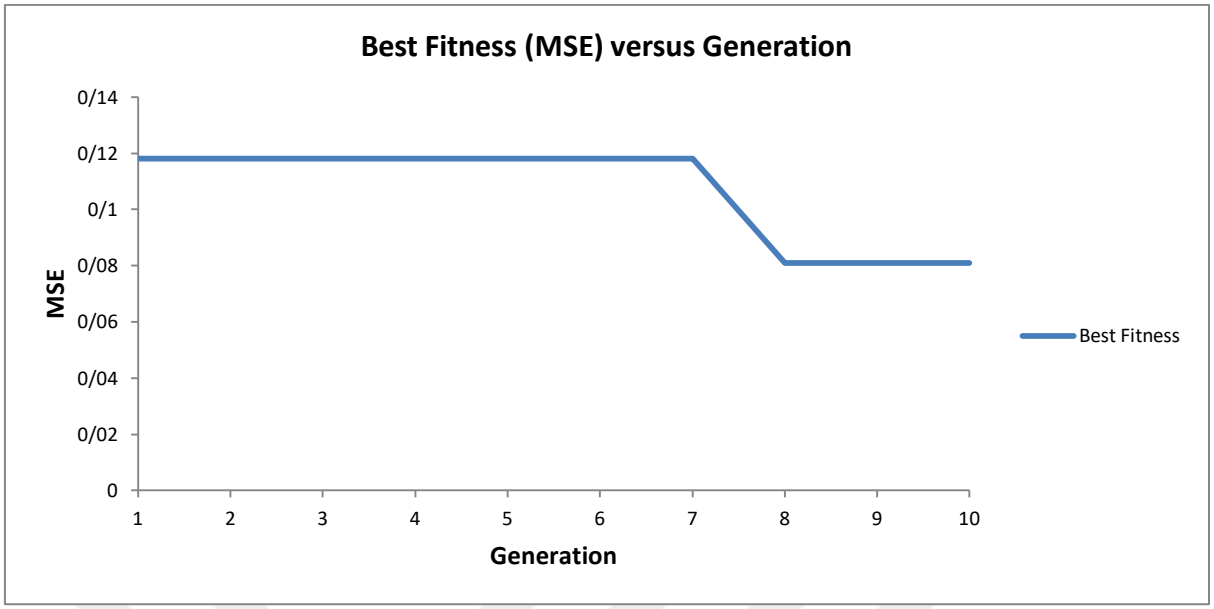
Optimization Summary	Best Fitness	Average Fitness
Generation #	12	12
Minimum MSE	0/053741567	0/077448376
Final MSE	0/053741567	0/078197271

Tablo 4.45. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

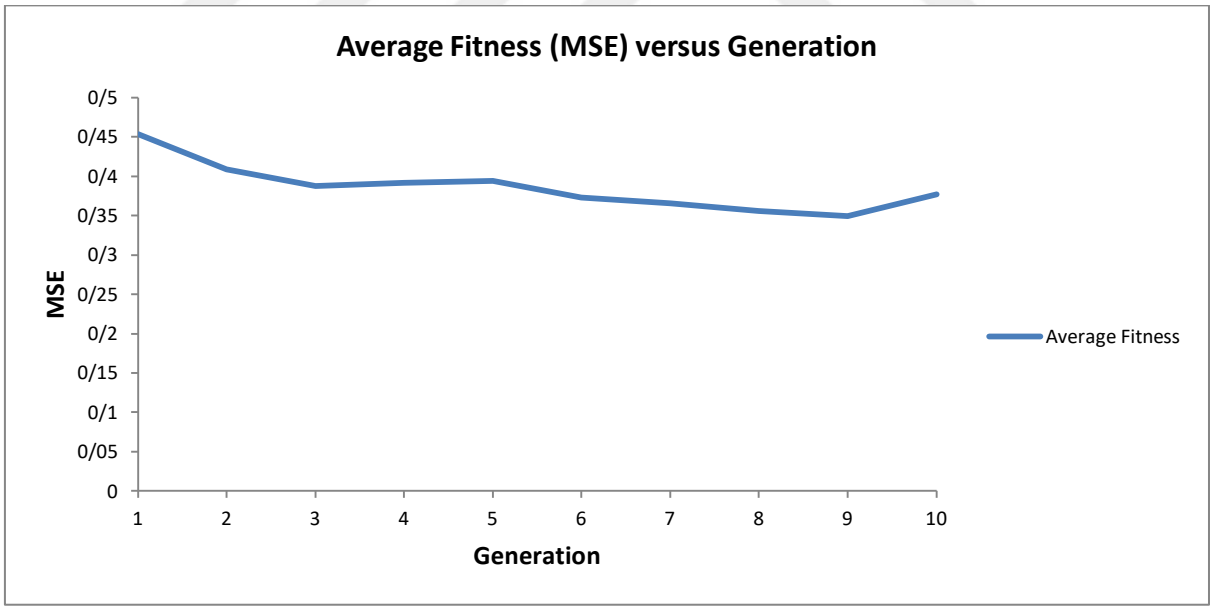
RMSE	MSE	MAE	r
0/279929	0/07836	0/27201	0/8524

Şekil 4.52, 4.53 ve Tablo 4.43, 4.44'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı onikinci nesilde ve ortalama en düşük hata oranı onikinci nesilde eğitim verilerinde sırasıyla 0/05374157ve 0/07744838 elde edildi.Ve test verileri için Tablo 4.45'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0/8524, 0/27201, 0/07836, 0/279929 elde edildi.

4.9.3.15. linear Tanh Axon Transfer Fonksiyonu Delta-Bar-Delta Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.54. Nesile karşı en iyi fitness (MSE)(linear Tanh axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)



Şekil 4.55. Nesile karşı ortalama fitness (MSE)(linear Tanh axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

Tablo 4.46. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(linear Tanh axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/11798545	0/45314372
2	0/11798545	0/40880051
3	0/11798545	0/38723642
4	0/11798545	0/39149898
5	0/11798545	0/3943325
6	0/11798545	0/37264809
7	0/11798545	0/36592111
8	0/08085932	0/35591164
9	0/08085932	0/34945798
10	0/08085932	0/37680024

Minimum Best Fitness	Minimum Average Fitness
0/08085932	0/34945798

Tablo 4.47. Optimizasyon özeti(Eğitim verileri için)(linear Tanh axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

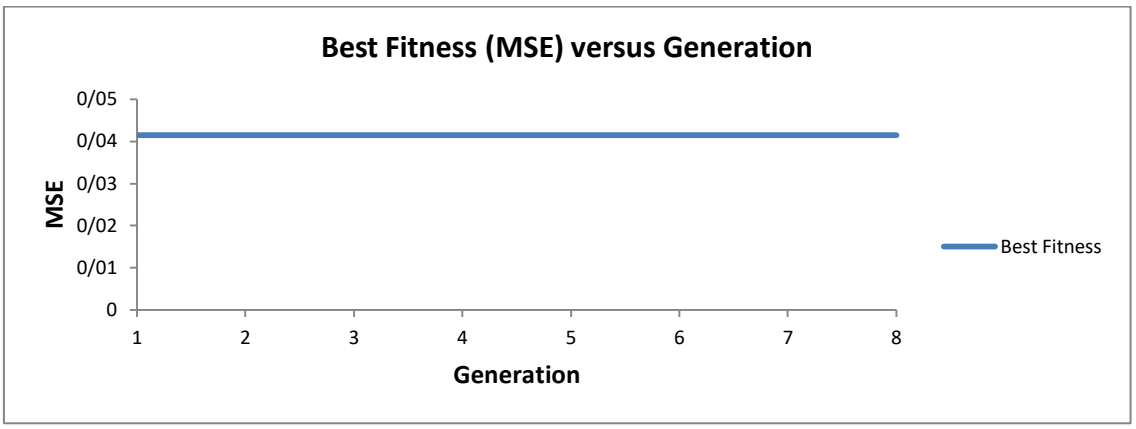
Optimization Summary	*Best Fitness*	*Average Fitness*
Generation #	8	9
Minimum MSE	0/080859318	0/349457979
Final MSE	0/080859318	0/376800239

Tablo 4.48. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için) (linear Tanh axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

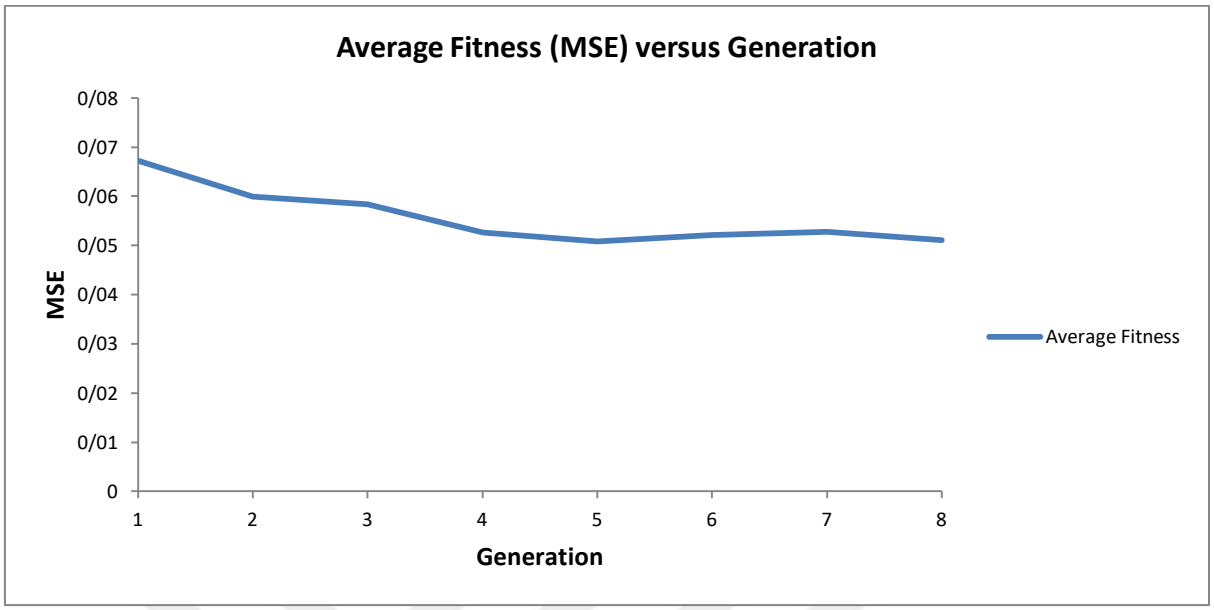
RMSE	MSE	MAE	r
0/177229	0/03141	0/14978	0

Şekil 4.54, 4.55 ve Tablo 4.46, 4.47'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı sekizinci nesilde ve ortalama en düşük hata oranı dokuzuncu nesilde eğitim verilerinde sırasıyla 0/08085932 ve 0/34945798 elde edildi.Ve test verileri için Tablo 4.48'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0, 0/14978, 0/03141, 0/177229 elde edildi.

4.9.3.16. linear Sigmoid Axon Aransfer Fonksiyonu Delta-Bar-Delta Öğrenme Kuralı ile Çok Katmanlı Perceptronı



Şekil 4.56. Nesile karşı en iyi fitness (MSE)(linear Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)



Şekil 4.57. Nesile karşı ortalama fitness (MSE)(linear Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

Tablo 4.49. Nesile karşı en iyi fitness ve ortalama fitnes (MSE)(linear Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

Nesil	Best Fitness	Average Fitness
1	0/04146234	0/06720027
2	0/04146234	0/05990904
3	0/04146234	0/05836456
4	0/04146234	0/0526071
5	0/04146234	0/05087283
6	0/04146234	0/05217611
7	0/04146234	0/05276207
8	0/04146234	0/05105398
	Minimum Best Fitness	Minimum Average Fitness
	0/04146234	0/05087283

Tablo 4.50. Optimizasyon özeti (Eğitim verileri için)(linear Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

Optimization Summary	Best Fitness	Average Fitness
Generation #	1	5
Minimum MSE	0/041462343	0/050872825
Final MSE	0/041462343	0/051053979

Tablo 4.51. Sonuçların istatistiksel parametrelerle karşılaştırılması(test verileri için)(linear Sigmoid Axon transfer fonksiyonu Delta-Bar-Delta öğrenme kuralı)

RMSE	MSE	MAE	r
0/143736	0/02066	0/12596	0

Şekil 4.56, 4.57 ve Tablo 4.49, 4.50'te gösterildiği gibi, eğitim verilerinde (MSE) en düşük hata oranı İlk nesilde ve ortalama en düşük hata oranı beşinci nesilde eğitim verilerinde sırasıyla 0/04146234 ve 0/05087283 elde edildi.Ve test verileri için Tablo 4.51'da gösterildiği gibi "r"(Linear Correlation Coefficient), "MAE"(Mean absolute error)," MSE "(Mean Square Error),"RMSE"(Root Mean Square Error) değerleri sırasıyla 0, 0/12596, 0/02066, 0/143736 elde edildi.

4.9.4. Yapay Sinir Ağı Sonuçlarının Değerlendirilmesi

Bu bölümde, sonuçları karşılaştırmak için RMSE[1], MAE[2] ve r[1]'nin üç istatistiksel göstergeleri kullanıldı ki aşağıdaki gibi hesaplanabilir.

[1] Root Mean Square Error
[2] Mean absolute error

$$\mathrm{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(yi-ti)\hat{}2}{n}} \quad (4.3)$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |yi - ti| \quad (4.4)$$

$$r = \frac{\sum_{i=1}^{n}(yi-\bar{y})*(ti-\bar{t})}{\sqrt{\sum_{i=1}^{n}(yi-\bar{y})\hat{}2*(ti-\bar{t})\hat{}2}} \quad (4.5)$$

Burada:

yi : yapay sinir ağından i'inci numunenin tahmin edilen değeri

$\bar{y}$: yapay sinir ağından alınan örneklerin tahmini ortalama değeridir

t_i: i'inci ölçülen değeri

$\underline{t}$: ölçülen ortalama değeri

n: toplam örnek sayısı

RMSE ve MAE ne kadar düşükse, yapay sinir ağının performansı o kadar iyidir. "r" Korelasyonuna, -1'den + 1'e değiştirin. −1'lik bir korelasyon, mükemmel bir negatif korelasyonu gösterirken, 1'lik bir korelasyon, mükemmel bir pozitif korelasyonu gösterir. 0 değerindeki bir korelasyon, iki değişkenin hareketleri arasında sıfır veya hiçbir ilişki olmadığını gösterir.

Tablo 4-52, farklı transfer fonksiyonları ile ve farklı öğrenme kuralları ile yapay sinir ağı uygulamasının sonuçlarının genel bir karşılaştırmasını göstermektedir.

Tablo 4-52'de gösterildiği gibi Momentum öğrenme kuralı ve Tanh transfer fonksiyonu ile sırasıyla 0.185 ve 0.171 değerlerine sahip "RMSE" ve "MAE" değerleri en düşük hata değerlerini ve "r" -0.95 değeri, giriş verileri ile tahmin edilen sonuçlar arasındaki en yüksek korelasyonu gösterir. Ve Conjugate Gradient öğrenme kuralı ve Sigmoid transfer fonksiyonu ile sırasıyla 0.163 ve 0.145 değerlerine sahip "RMSE" ve

[1] Linear Correlation Coefficient

"MAE" değerleri en düşük hata değerlerini ve "r" 0,74 değeri, giriş verileri ile tahmin edilen sonuçlar arasındaki en yüksek korelasyonu gösterir. Quickprop kuralında ve Tanh transfer fonksiyonunda, sırasıyla 0.166 ve 0.147 değerlerine sahip "RMSE" ve "MAE" değerleri en düşük hata değerlerini ve Sigmoid transfer fonksiyonunda"r"- 0. 87 değeri, giriş verileri ile tahmin edilen sonuçlar arasındaki en yüksek korelasyonu gösterir. DeltaBarDelta kuralında ve LinearSigmoid transfer fonksiyonunda, sırasıyla 0.177 ve 0.150 değerlerine sahip "RMSE" ve "MAE" değerleri en düşük hata değerlerini ve Tanh transfer fonksiyonunda"r" -0,564 değeri, giriş verileri ile tahmin edilen sonuçlar arasındaki en yüksek korelasyonu gösterir.

Genel olarak, "RMSE" ve "MAE" nin düşük hata oranı ve giriş verileri ile tahmin edilen sonuçlar arasındaki yüksek derecede "r" korelasyonu nedeniyle, Tanh transfer fonksiyonu ile Momentum öğrenme kuralı ve Sigmoid transfer fonksiyonu ile Conjugate Gradiant öğrenme kuralı, diğer yöntemlerden daha iyi sonuç verir. Ayrıca Momentun öğrenme kuralı ve Tanh transfer fonksiyonu ile optimizasyon diyagramına göre (Şekil 4.25) minimum hatanın onuncu nesilde elde edildiği görülmektedir.

Tablo 4.52. Tüm sonuçların istatistiksel parametrelerle karşılaştırılması (test verileri için)

Number	Learning Rule	Transfer fonksiyon	RMSE	MSE	MAE	r
1	Momentun	Tanh	0/185175592	0/03429	0/17077	-0/95261
2	Momentun	Sigmoid	0/237212984	0/05627	0/22267	-0/6753
3	Momentun	Linear Tanh	0/215336945	0/04637	0/19623	0/32268
4	Momentun	LinearSigmoid	0/213658606	0/04565	0/19888	0/05686
5	Conjugate Gradiant	Tanh	0/161895028	0/02621	0/14512	-0/12682
6	Conjugate Gradiant	Sigmoid	0/163431943	0/02671	0/1452	0/73705
7	Conjugate Gradiant	Linear Tanh	0/218746429	0/04785	0/20658	0/17583
8	Conjugate Gradiant	LinearSigmoid	0/211919796	0/04491	0/18292	0/28192
9	Quickprop	Tanh	0/165620047	0/02743	0/14721	-0/31479
10	Quickprop	Sigmoid	0/301496269	0/0909	0/29094	-0/86952
11	Quickprop	Linear Tanh	0/171639156	0/02946	0/15714	0/26478
12	Quickprop	LinearSigmoid	0/218265893	0/04764	0/19692	0/288
13	DeltaBarDelta	Tanh	0/819310686	0/67127	0/2257	-0/56396
14	DeltaBarDelta	Sigmoid	0/279928562	0/07836	0/27201	0/8524
15	DeltaBarDelta	Linear Tanh	0/224499443	0/0504	0/2106	0/3128
16	DeltaBarDelta	LinearSigmoid	0/177228666	0/03141	0/14978	0

5. SONUÇLAR VE ÖNERİLER

5.1. Sonuçlar

Son zamanlarda, kentleşme ve sanayileşmedeki artış ve su kirliliği geniş alanlar için bir tehdit haline gelmiştir. Bunun sonucunda iyi ve verimli tasarlanmış bir izleme ağı, su kalitesi problemlerini ve uzun vadeli eğilimleri belirlemede çok önemli bir ön koşul haline gelmiştir. (Dixon et al., 1999). Bir nehirdeki su kalitesi izleme istasyonları ağının iyi ve verimli bir şekilde tasarlanmasının amacı, nehir suyu kalitesinin izlenmesi hedeflerine ve mevcut kısıtlama türlerine (operasyonel, sosyal, idari, yasal vb. Kısıtlamalar) göre nehir suyunun kalitesi ve miktarı hakkında en fazla bilgiyi elde etmektir. Aynı zamanda, uygun maliyetli tasarımı sağlamak için ekonomik hususlar nehir suyu kalitesi izleme istasyonları ağının verimli tasarımına dahil edilmelidir. Bu nedenle, nehir suyu kalitesi izleme istasyonları ağının verimli tasarımı sorunu bir optimizasyon sorunudur.Çünkü bu tasarımda nehir suyunun kalitesi ve miktarı hakkında en fazla bilgiyi elde etmemiz ve aynı zamanda bu tasarımın maliyetlerini en aza indirmemiz gerekir. Optimizasyon için güçlü özellikleri nedeniyle GA'lar hala ilk tercihtir. Bu yöntem kullanılarak birçok çalışma yapılmıştır. GA'lar optimal bölge bulma yeteneğini gösterdiler.Ancak bu çalışmalar MOGA 'ları doğrudan CBS ile birleştirmedi. Bu çalışma, GA'nın sonuçlarını web haritasında otomatik olarak göstermek için bağımsız bir web tabanlı uygulamada MOGA "NSGA-II"yi açık kaynak araçlarıyla GIS ile başarılı bir şekilde entegre edebildi. Önceki çalışmalarda bu entegrasyon, masaüstü tabanlı yazılımlarda MOGA'nın tek amaçlı GA'ya dönüştürülmesi veya MATLAB gibi ticari yazılımlar kullanılarak yapılmıştır. Açık kaynaklı araçların kullanımı ve bu uygulamanın web tabanlı yapısı, bu uygulamaya kolay ve ücretsiz erişim sağlar. Önceki çalışmalarda nehir suyu kalitesi izleme istasyonları ağı tasariminda sadece nehir kalitesi izleme hedefleri göz önünde bulundurulmuştur. Bu çalışmada, önceki çalışmalardan farklı olarak, aynı zamanda eğim ve aylık ortalama sıcaklık faktörlerine göre hem su kalitesi izleme hedefleri hem de su kalitesi izleme istasyonlarının kurulabileceği uygun yerler ele alınmıştır. Bu da NSGA-II algoritması tarafından bulunan optimal

konumların sonuçlarını daha doğru ve gerçeklikle tutarlı hale getirir, ve su kalitesi izleme hedefleri açısından uygun ancak kısıtlamalar nedeniyle su kalitesi izleme istasyonu kurulması mümkün olmayan yerlerin bulunmasından kaçınmaktır. NSGA-II sonuçları ile Analitik Hiyerarşik süreç ve Bulanık Mantık yöntemlerinin sonuçları karşılaştırıldığında, bu yöntemlerin sonuçlarının büyük ölçüde uyumlu olduğu gözlemlendi.

Bu çalışmada, NSGA-II algoritmasından elde edilen Pareto optimal cephesi incelendikten sonra, dikey eksende Pareto optimal cephenin üst kısmında bu algoritmanın sadece zayıf Pareto optimal çözümlerini bulduğu gözlemlenmiştir. Bunun nedeni, Pareto optimal çözümlerini bulma koşullarının katı olması ve NSGA-II algoritması da dahil olmak üzere birçok MOO algoritmasının Pareto optimal çözümlerini garanti edememesi, sadece zayıf Pareto optimal çözümleri üretmesidir. Sonuç olarak bu çalışmada, su kalitesi izleme istasyonlarının optimal yerleşimi için nihai çözüm olarak zayıf Pareto optimal çözümlerin dışında Pareto optimal cephedeki diğer çözümler seçildiler.

Bir sonraki adımda, nehir suyu kalitesi izleme istasyonları optimal yerlerin bulunmasını hızlandırmak için GA ile optimize edilmiş çok katmanlı Perceptron sinir ağı (MLP) kullanıldı. Bu aşamada NSGA-II algoritmada kullanılan veriler MLP sinir ağı giriş verisi olarak kullanıldı. Ayrıca, bu yapay sinir ağını eğitmek için NSGA-II algoritmalardan elde edilen su kalitesi izleme istasyonlarının optimal yerleri kullanıldı. Bu çalışmada, verilerin% 70'inde yapay sinir ağı eğitimi,% 20'sinde yapay sinir ağı testi ve% 10'unda cross validation için kullanıldı. Ayrıca 1- Tanh 2- Sigmoid 3-Linear Tanh 4-Linear Sigmoid Transfer Fonksiyonları ve 1-Momentum 2- Conjugate Gradient 3- Quickprop 4-Delta-Bar-Delta öğrenme kuralları kullanıldı. Son olarak bu yapay sinir ağından elde edilen sonuçlar "RMSE", "MAE" ve "r" istatistiksel göstergeleriyle değerlendirildi. Genel olarak, "RMSE" ve "MAE" nin düşük hata oranı ve giriş verileri ile tahmin edilen sonuçlar arasındaki yüksek derecede "r" korelasyonu nedeniyle, Tanh transfer fonksiyonu ile Momentum öğrenme kuralı (RMSE = 0.185, MAE=0.171,r=-0.95) ve Sigmoid transfer fonksiyonu ile Conjugate Gradiant öğrenme kuralı (RMSE = 0.163, MAE=0.0.145,r=0.74), diğer yöntemlerden daha iyi sonuç verdiler.

5.2. Öneriler

1-İleri çalışmalarda, bu web uygulamada bir YSA entegre etmesi önerilir. Bu yaklaşımın arkasındaki mantık, nehir havzasındaki amaç fonksiyonları tarafından kullanılan verileri güncellenmesi gerektiğinde, kullanıcıların GA algoritmasını yeniden çalıştırmadan optimal konumları yeniden belirlemelerini sağlamaktı. "Öğrenilen" ağırlıkların belirli bir doğruluk düzeyinde geçerli olabilmesi için bunun sadece aynı nehir havzası içinde geçerli olduğu varsayılmaktadır.

2- MO problemlerinde, optimal çözümlerin seçiminde Pareto kavramının önemli bir rol oynadığı göz önüne alındığında, gelecekteki çalışmalarda, NSGA-II algoritmasından elde edilen Pareto optimal cephesi ile çalışma alanında elde edilen optimal konumlar arasında doğrudan bağlantı önerilmektedir. Sonuç olarak karar verici, Pareto optimal cephesini ve harita üzerindeki konumlarını aynı anda inceleyerek, nihai çözümleri seçebilecektir.

3- Bu çalışmada, nehir suyu izleme ağını tasarlamak için genel amaçlar kullanılmıştır; Belirli uygulamalara sahip çalışmalar için bir nehir suyu izleme ağı tasarlamak için başka amaçların kullanılması tavsiye edilir.

4- Bu çalışmada, sorunun cevaplarını sınırlamak için sadece eğim ve sıcaklık kısıtları kullanıldı. Sorunun cevaplarını sınırlamak için diğer kısıtlamaların kullanılması önerilir.

5- Bir su kalitesi izleme istasyonları ağı tasarlamak için karınca kolonisi algoritması ve tabu arama yöntemi gibi diğer meta-sezgisel yöntemlerinde kullanılması önerilir.

6. KAYNAKLAR

Abo-Hamad, W., & Arisha, A., 2011. "Simulation-optimisation methods in supply chain applications: a review." Irish Journal of Management 30.2, 95.

Anderson, D., & McNeill, G., 1992. "Artificial neural networks technology." *Kaman Sciences Corporation* 258.6, 1-83.

Ahmadipari, M., Hoveidi, H., Jafari, H. R., & Pazoki, M., 2018. An integrated environmental management approach to industrial site selection by genetic algorithm and fuzzy analytic hierarchy process in geographical information systems. Global Journal of Environmental Science and Management, 4(3), 339-350.

Azzini, A., & Tettamanzi, A., 2006 "A new genetic approach for neural network design and optimization." PhD, University of Milan, Milan.

Bačanin Džakula, N., 2020. "Multi-Layer Perceptron Training by Genetic Algorithms." Sintez International Scientific Conference on Information Technology and Data Related Research. Singidunum University.

Bäck, T., 1993. Optimal mutation rates in genetic search. In: Proceedings of the fifth international conference on genetic algorithms.

Ballatore, A. Tahir, A. McArdle, G. ve Bertolotto, M., 2011. "A comparison of open source geospatial technologies for web mapping", International Journal of Web Engineering and Technology, 6: 354-374.

Banai-Kashani, R., 1989. A New Method for Site Suitability Analysis: The Analytic Hierarchy Process, Environmental Management, Vol. 13 (6), pp: 685-693.

Bahiraei, M., Nazari, S., Moayedi, H., & Safarzadeh, H., 2020. Using neural network optimized by imperialist competition method and genetic algorithm to predict water productivity of a nanofluid-based solar still equipped with thermoelectric modules. Powder Technology, 366, 571-586.

Bai, Q., Labi, S., & Sinha, K. C., 2012. "Trade-off analysis for multiobjective optimization in transportation asset management by generating Pareto frontiers using extreme points nondominated sorting genetic algorithm II." Journal of Transportation Engineering 138.6, 798-808.

Bolc,L.&Cytowski,J., 1992. Search Methods for Artificial Intelligence,London:Academic Press.

Booker,L., 1987. Improving search in genetic algorithms, in Davis, L ,ed. Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, Los Altos, CA.

Bui, D. T., Tuan, T. A., Klempe, H., Pradhan, B., & Revhaug, I., 2015. Spatial prediction models for shallow landslide hazards: acomparative assessment of the efficacy of support vector machines, artificial neural networks, kernel logistic regression, and logistic model tree. Landslides13(2),361–378.

Boussaïd, I., Lepagnot, J. and Siarry, P., 2013. A survey on optimization metaheuristics, Information Sciences, 237, 82-117.

Bui,K.-T.T., Bui, D.T., Zou,J., Doan, C.V., Revhaug,I., 2016. A novel hybrid artificial intelligent approach based on neural fuzzy inference model and particle swarm optimization for horizontal displacement modeling of hydropower dam. Neural Comput.& Applic.27(8).

Burnaz, S., and Topcu, Y.I., 2006. A Multiple-Criteria Decision-making Approach for the Evaluation of Retail Location, Journal of Multi-Criteria Decision Analysis, Vol. 14, pp: 67-76.

Burns, E., Lemons, S., Ruml, W. and Zhou, R., 2010. Best-first heuristic search for multicore machines, Journal of Artificial Intelligence Research, 39, 689-743.

Censor, Y., 1977. Pareto optimality in multiobjective problems, Applied Mathematics and Optimization, 4, 1, 41-59.

Chicco, Gianfranco, and Andrea Mazza., 2020. "Metaheuristic Optimization of Power and Energy Systems: Underlying Principles and Main Issues of the 'Rush to Heuristics'." Energies 13.19, 5097.

Cömert, Ç., & Akinci, H.., 2003 Web services: an e-government perspective.

CÖMERT, C., 2004. Web services and national spatial data infrastructure (NSDI). In: Proceedings of Geo-Imagery Bridging Continents, XXth ISPRS Congress. Turkey Commission.

Akinci, H., & Cömert, Ç., 2007. Geoportals and their role in spatial data infrastructures. Department of Geodesy and Photogrammetry Engineering, Turkey.

Cömert, Ç., Ulutaş, D., Akıncı, H., & Kara, G., 2008. Integrated coastal management, marine spatial data infrastructures, and semantic web services.

Casillas, M. V., Puig, V., Garza-Castanón, L. E., & Rosich, A., 2013. Optimal sensor placement for leak location in water distribution networks using genetic algorithms. Sensors, 13(11), 14984-15005.

Castillo, P. A., Merelo, J. J., Prieto, A., Rivas, V., & Romero, G., 2000. G-Prop: Global optimization of multilayer perceptrons using GAs. *Neurocomputing*, *35*(1-4), 149-163.

Cavanagh, N., R.N. Nordin, L.W. Pommen and L.G. Swain., 1998. Guidelines for Designing and Implementing a Water Quality Monitoring Program in British Columbia. Ministry of Environment, Lands and Parks. Province of British Columbia.

Cheng, E.W.L., and Li, H., 2001. Information Priority-setting for Better Resource Allocation Using Analytic Hierarchy Process, Information Management and Computer Security, Vol. 9, No. 2, pp.61–70.

Chicco, G., & Mazza, A., 2020. "Metaheuristic Optimization of Power and Energy Systems: Underlying Principles and Main Issues of the 'Rush to Heuristics'." Energies 13.19: 5097.

Correa, A., Gonzalez, A., & Ladino, C., 2011. Genetic algorithm optimization for selecting the best architecture of a multi-layer perceptron neural network: a credit scoring case. SAS Global Forum. 2011.

Coello, C. A. C., & Lamont, G. B. (Eds.)., 2004 . Applications of Multi-Objective Evolutionary Algorithms (Vol. 1). World Scientific.

Chen, C-F., 2006. Applying the Analytical Hierarchy Process (AHP) Approach to Convention Site Selection, Journal of Travel Research, Vol. 45, pp: 167-174.

Chicco, G., & Mazza, A., 2020 "Metaheuristic Optimization of Power and Energy Systems: Underlying Principles and Main Issues of the 'Rush to Heuristics'." Energies 13.19: 5097.

Davies, M., 2001. Adaptive AHP: AReviewof Marketing Applications with Extensions, European Journal of Marketing,Vol. 35, Nos. 7/8, pp.872–893.

Deb, K., 2001. Multi-objective optimization using evolutionary algorithms, vol. 16, John Wiley & Sons.

Dev,K., 2011. "Multi-objective optimisation using evolutionary algorithms: an introduction." *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, London, 3-34.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T., 2002 .A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. Evolutionary Computation, IEEE Transactions on, 6(2), 182-197.

Dev,K., 2011. Multi-objective optimization using evolutionary algorithms. Vol. 16. John Wiley & Sons.

Dev,K., 2011. "Multi-objective optimization using evolutionary algorithms: an introduction." KanGAL Report 2011003.

De Jong, K. A., 1975 .Analysis of the behavior of a class of genetic adaptive systems.

De Jong, K., 1980. Adaptive system design: a genetic approach. IEEE Transactions on Systems, Man, and Cybernetics, 10.9: 566-574.

Dev,K., 1995 .Optimization for Engineering Design: Algorithms and Examples, NewDelhi: Prentice-Hall.

Deb K., 2002. Agrawal S, Pratap A, Meyarivan T. A fast and Elitist multi-objective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation; 6(2):182–197

Delipetrev, B., Jonoski, A. & Solomatine, D., 2014. Development of a web application for water resources based on open source software, Computers & Geosciences. 62, pp. 35–42.

Ducheyne, E. I., De Wulf, R. R., & De Baets, B., 2006. A spatial approach to forest-management optimization: linking GIS and multiple objective genetic

algorithms. International Journal of Geographical Information Science, 20.8: 917-928.

Du, K., L. and Swamy, M., N., S., 2016. Search and optimization by metaheuristics, Techniques and Algorithms Inspired by Nature, Birkhauser: Basel, Switzerland.

Dixon, W., Chiswell, B., 1996. Review of aquatic monitoring program design. Water Res. 30, 1935–1948.

Dixon, W., Smyth, G.K., Chiswell, B., 1999. Optimized selection of river sampling sites. Water Res. 33, 971–978.

Esterby, S. R., El-Shaarawi, A. H., & Block, H. O., 1992. Detection of water quality changes along a river system. Environ. Monit Assess. 23, 219–242.

Eiben, Agoston E., and James E. Smith., 2003. "Genetic algorithms." Introduction to Evolutionary Computing. Springer, Berlin, Heidelberg, 37-69.

Ehrgott, M., & Wiecek, M. M., 2005 .Mutiobjective Programming. In Multiple criteria Decision Analysis: State of the art surveys (pp. 667-708). Springer New York.

Ehsan, R. Muhammad, Sishaj P. Simon, and P. R. Venkateswaran., 2017. "Day-ahead forecasting of solar photovoltaic output power using multilayer perceptron." Neural Computing and Applications 28.12, 3981-3992.

Eldr andaly, K., Eldin, N., and Sui, D., 2003. A COM-based Spatial Decision Support System for Industrial Site Selection, Journal of Geographic Information and Decision Analysis, Vol. 7, No. 2, pp: 72-92.

Fahlman, S.E., 1998. An Empirical Study of Learning Speed in Back-propagation Networks. Technical Report CMU-CS-88–162. Carnegie Mellon University, Pittsburgh, PA, pp. 15213.

Ferentinos, K. P., & Tsiligiridis, T. A., 2007. "Adaptive design optimization of wireless sensor networks using genetic algorithms." Computer Networks 51.4, 1031-1051.

Fogel, Lawrence J., Alvin J. Owens, and Michael J. Walsh., 1996. Artificial intelligence through simulated evolution.

Fotheringham, Stewart, Peter Rogerson., 2005. Spatial Analysis and GIS. United Kingdom: Taylor & Francis. ch. 10.

Fustes, D., Cantorna, D., Dafonte, C., Arcay, B., Iglesias, A., & Manteiga, M., 2014. A Cloud-integrated Web Platform for MarineMonitoring Using GIS and Remote Sensing. Application to oil Spill Detection Through SAR Images. Future Generation computer Systems, vol. 34, pp. 155-160.

Gen, M. & Cheng, R., 1997. Genetic Algorithms and Engineering Design. New York: John Wiley & Sons.

Gen, M., Cheng, R., & Lin, L., 2008. Network models and optimization: Multiobjective genetic algorithm approach. Springer Science & Business Media.

GeoServer., 2015. "About GeoServer ", http://geoserver.org/about/. 10 Ekim.

Goudjil, K., & Sbartai, B., 2007. Optimization of shear wave velocity (Vs) from a post-liquefaction settlement using a genetic algorithm Multi-Objective NSGA II.

Guo, H. Y., Zhang, L., Zhang, L. L., & Zhou, J. X., 2004. Optimal placement of sensors for structural health monitoring using improved genetic algorithms. Smart materials and structures, 13(3), 528.

Handayanto, R. T., Gunarti, A. S., & Seta Samsiana, H., 2015. A Web-GIS based integrated optimum location assessment tool for gas station using genetic algorithms. ARPN J. Eng. Appl. Sci. 10.3 (2015): 1383-1388.

Hämäläinen, R. P., & Seppäläinen, T. O., 1986. The analytic network process in energy policy planning. Socio-Economic Planning Sciences, 20.6: 399-405.

Harmancioglu, N.B., Alpaslan, M.N., 1992. Basic approaches in design of water quality monitoring networks. Water Sci. Technol. 30, 49–56.

Holland, J., 1992. Adaptation in Natural and Artificial System, Ann Arbor: University of Michigan Press; 1975, MA: MIT Press.

Holland, John H., 1962. Outline for a logical theory of adaptive systems. Journal of the ACM (JACM) 9.3: 297-314.

Hudak, P.F., Loaiciga, H.A., Marino, M.A., 1995. Regional-scale ground water quality monitoring via integer programming. J. Hydrol. 164, 153–170.

Url, 1. http://www.neurosolutions.com/neurosolutions/help. 20 Ağustos 2021.

Url, 2. https://sop.tik.ee.ethz.ch/pisa/?page=selvar.php

Url, 3. https://github.com/abfo/shapefile/tree/master/Catfood.Shapefile

Url, 4. https://wiki.openstreetmap.org/wiki/Openlayers_POI_layer_example

Url, 5. https://wiki.openstreetmap.org/wiki/Points_of_interest

Ilbeigi, M., Ghomeishi, M., & Dehghanbanadaki, A., 2020. Prediction and optimization of energy consumption in an office building using artificial neural network and a genetic algorithm. Sustainable Cities and Society, 61, 102325.

Icaga, Y., 2005. "Genetic algorithm usage in water quality monitoring networks optimization in Gediz (Turkey) river basin." Environmental Monitoring and Assessment 108.1: 261-277.

Ishibuchi H, Tsukamoto N, Nojima Y., 2008. Evolutionary many objective optimization: a short review. In: Proceedings of the IEEE congress on evolutionary computation, pp 2419–2426.

Inoue, M., & Yamamoto, K., 2013. Method for evaluating the location of tourist-related public facilities using genetic algorithms and GIS. Journal of Communication and Computer, 10.4: 496-512.

J. Feldman and R. Rojas., 1996. Neural Networks: A Systematic Introduction. Springer Berlin Heidelberg.

Jha, M. K., & Schonfeld, P., 2000. Integrating genetic algorithms and geographic information system to optimize highway alignments. Transportation Research Record, 1719(1), 233-240.

Jha, M. K., Schonfeld, P., & Samanta, S., 2007. Optimizing rail transit routes with genetic algorithms and geographic information system. Journal of Urban Planning and Development, 2007, 133.3: 161-171.

Jun, C., 2000. Design of an Intelligent Geographic Information System for Multicriteria Site Analysis, URISA Journal 12 (3); pp: 5-17.

Kavzoglu,T.,Mather,P.M., 2003. The use of back-propagation artificial neural networks in land cover classification. Int.J.RemoteSens. 24,4907–4938.

Ki, Junghoon., 2011. Developing a Geospatial web-GIS System for andscape and Urban planning. Int. Journal of Digital Earth, vol. 6, no. 6, pp. 580-588.

Kim, J. H. & Myung, H., 1996. A two-phase evolutionary programming for general constrained optimization problem, Proceeding of the 1st Annual Conference on Evolutionary Programming

Lettenmaier, D.P., 1979. Dimensionality problems in water quality network design. Water Resour. Res. 15, 1692–1700

Le Berre, M., Hnaien, F., & Snoussi, H., 2011. Multi-objective optimization in wireless sensors networks. In: ICM 2011 Proceeding. IEEE, p. 1-4.

Lettenmaier, D.P., 1979. Dimensionality problems in water quality network design. Water Resour. Res. 15, 1692–1700.

Li B, Li J, Tang K, Yao X., 2015. Many-objective evolutionary algorithms: a survey. ACM Comput Surv 48(1):13

Liepins, G., Hilliard, M., Richardson, J. & Pallmer, M., 1990. Genetic algorithm application to set covering and traveling salesman problems, in Brown ed. OR/AI: The Integration of Problem Solving Strategies.

Liebetrau, A.M., 1979. Water quality sampling: some statistical considerations. Water Resour. Res. 15, 1717–1725.

Liepins, G. & Potter, W., 1991. A genetic algorithm approach to multiple faultdiagnosis, Davis ed. Handbook of Genetic Algorithms, New York: Van Nostrand Reinhold, 237–250.

Li B, Li J, Tang K, Yao X., 2015. Many-objective evolutionary algorithms: a survey. ACM Comput Surv 48(1):13

Li, Wei., 2004. "Using genetic algorithm for network intrusion detection." *Proceedings of the United States department of energy cyber security group* 1,1-8.

Lin, W. Y., Lee, W. Y., & Hong, T. P., 2003. Adapting crossover and mutation rates in genetic algorithms. J. Inf. Sci. Eng, 19.5: 889-903.

Li, X., & Yeh, A. G. O., 2005. Integration of genetic algorithms and GIS for optimal location search. International Journal of Geographical Information Science, 19.5: 581-601.

Li, Z., Zhong, R. Y., Barenji, A. V., Liu, J. J., Yu, C. X., & Huang, G. Q., 2019. Bi-objective hybrid flow shop scheduling with common due date. Operational Research, 1-26.

Liu, W., Gao, W. C., Sun, Y., & Xu, M. J., 2008. Optimal sensor placement for spatial lattice structure based on genetic algorithms. Journal of Sound and Vibration, 317(1-2), 175-189.

Loftis, J.C., McBride, G.B., Ellis, J.C., 1991. Considerations of scale in water quality monitoring and data analysis. Water Resour Bull. 27, 255–264.

Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z., 2019. Genetic algorithm approach to design of multi-layer perceptron for combined cycle power plant electrical power output estimation. Energies, 12(22), 4352.

K. Gurney. An Introduction to Neural Networks., 2003. Taylor & Francis.

K. Miettinen., 1998. Nonlinear Multiobjective Optimization. Kluwer Academic Publishers.

Kanal, L. and Kumar, V. 2012. Search in artificial intelligence, Springer Science & Business Media. Kanal, L. and Kumar, Search in artificial intelligence, Springer Science & Business Media.

Kim, S.C. and Min, K.J., 2004. Determining Multi-criteria Priorities in the Planning of Electric Power Generation:The Development of an Analytic Hierarchy Process for Using the Opinions of Experts, International Journal of Management, Vol. 21, No. 2, pp.186–193.

Kosko, Bart., 1994. "Fuzzy systems as universal approximators." IEEE transactions on computers 43.11, 1329-1333.

Kuruüzüm, A. ve Atsan, N., 2001. Analitik Hiyerarşi Yöntemi ve İşletmecilik Alanındaki Uygulamaları, Akdeniz,İ.İ.B.F. Dergisi, Sayı:1, 83-105.

Kulkarni, A.T., J. Hohanty, T.I. Eldho, E.P. Rao, B.K. Mohan., 2014. A Web-GIS Based Integrated Flood Assessment Modeling Tool for Coastal Urban Watersheds. Computer & Geosciences, vol. 64, pp. 7-14.

Maaranen, H., Miettinen, K., & Mäkelä, M. M., 2003. "Training multi layer perceptron network using a genetic algorithm as a global optimizer." *Metaheuristics: computer decision-making*. Springer, Boston, MA, 421-448.

Madetoja, E., Ruotsalainen, H., Monkkonen, V. M., Hamalainen, J., & Deb, K., 2008. Visualizing multi-dimensional Pareto-optimal fronts with a 3D virtual reality system. In 2008 International Multiconference on Computer Science and Information Technology (pp. 907-913). IEEE.

Marler, R., T. and Arora, J., S., 2010. The weighted sum method for multi-objective optimization: new insights, Structural and Multidisciplinary Optimization, 41, 6, 853- 862.

Michalewicz, Z., 1994. Genetic Algorithm + Data Structures = Evolution Programs. New York: Springer-Verlag.

Michalewicz, Z., 1995. A survey of constraint handling techniques in evolutionary computation methods, in McDonnell et al. eds. Evolutionary Programming IV, MA: MIT Press.

Michalewicz, Z., Dasgupta, D., Riche, R. G. L. & Schoenauer, M., 1996. Evolutionary algorithms for constrained engineering problems, Computers and Industrial Engineering, 30(4), 851–870.

Min, H., 1994. Location Analysis of International Consolidation Terminals Using the Analytic Hierarchy Process, Journal of Business Logistics, Vol. 15, No. 2, pp.25–44.

Miles, Eduardo J., 2009. Guidelines: Shallow Water Quality Monitoring Continuous Monitoring Station: Selection, Assembly & Construction.

Mirjalili, S., & Dong, J. S., 2020. Multi-objective optimization using artificial intelligence techniques. Springer.

Ning, S.K., Chang, N.B., 2002. Multi-objective, decision-based assessment of a water quality monitoring network in a river system. J. Environ. Manage. 4, 121–126.

Ohta, K., Kobashi, G., Takano, S., Kagaya, S., Yamada, H., Minakami, H., & Yamamura, E., 2007. Analysis of the Geographical Accessibility of Neurosurgical Emergency Hospitals in Sapporo City Using GIS and AHP, International Journal of Geographical Information Science, Vol. 21 (6), pp: 687-698.

Orvosh, D. & Davis, L., 1994. Using a genetic algorithm to optimize problems with feasibility constraints, Proceeding of the 1st IEEE Conference on Evolutionary Computation, 548–552.

Park, S. Y., Choi, J. H., Wang, S., & Park, S. S., 2006. Design of a water quality monitoring network in a large river system using the genetic algorithm. Ecological modelling 199.3: 289-297.

Peng, Bo, and Lei Li., 2015. "An improved localization algorithm based on genetic algorithm in wireless sensor networks." Cognitive Neurodynamics 9.2, 249-256.

PostGIS., 2015. "About PostGIS", http://postgis.net/. 10 Ekim.

Pohlheim, H., 2003. Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms. Genetic and Evolutionary Algorithm Toolbox. Hartmut Pohlheim. URL: http://www.geatbx.com/docu/algindex.html.

Pohlheim, H., 2006. GEATbx: Introduction Evolutionary Algorithms: Overview. Methods and Operators GEATbx version 3.

Puri, D., Borel, K., Vance, C., & Karthikeyan, R., 2017. "Optimization of a water quality monitoring network using a spatially referenced water quality model and a genetic algorithm." *Water* 9.9: 704.

Pham,B.T.,Bui,D.T.,Prakash,I.,Dholakia,M.B., 2016. Hybrid integration of Multilayer Perceptron Neural Networks and machine learning ensembles for landslide susceptibility assessment at Himalayan area (India) using GIS. Catena 149,52–63.

Ramanathan, R., 2001. A Note On the Use of the Analytic Hierarchy Process for Environmental Impact Assessment, Journal of Environmental Management, Vol. 63, pp: 27-35.

Ramezanian, R., Peymanfar, A., & Ebrahimi, S. B., 2019. "An integrated framework of genetic network programming and multi-layer perceptron neural network for prediction of daily stock return: An application in Tehran stock exchange market." Applied soft computing 82, 105551.

Raidl, G.R.& Julstrom, B.A., 2003. Edge Sets: An Effective Evolutionary Coding of Spanning Trees, IEEE Transactions on Evolutionary Computation, 7(3), 225–239.

Rechenberg, I., 1965 .Cybernetic solution path of an experimental problem.

Rechenberg, I., 1973. Optimieriung technischer Systeme nach Prinzipien der biologischen Evolution, Stuttgart: Frommann-Holzboog.

Saaty, T.L., 1980. The Analytic Hiearchy Process, New York, McGraw-Hill.

Saaty, T.L., 1991. Modelling the Graduate Business School Admission Process, Socio-Economic Planning Science, Vol. 25, pp.155–162.

Saaty, T.L., 1991. Some Mathematical Concepts of The Analytic Hierarchy Process, Behaviormetrika, No. 29, pp: 1- 9.

Saaty, T. L., and Vargas, Luis G., 2001. Models, Methods, Concepts & Applications of the Analytic Hierarchy Process. Springer, International Series in Operations Research & Management Science, Vol. 34, 352p.

Sakawa, M., 2002. Genetic algorithms and fuzzy multiobjective optimization. Springer Science & Business Media.

Sadowski,L.,Hola,J.,Czarnecki,S.,Wang,D., 2018. Pull-off adhesion prediction of variable thick overlay to the substrate. Autom.Constr.85,10–23. Schölkopf, B.,Smola, A.J.,2002. Learning with Kernels.

Schwefel, H., 1995. Evolution and Optimum Seeking, New York: Wiley & Sons.

Siddiqui, M.Z, Ever ett, J.W., and Vieux, B.E., 1996. Landfill Siting Using Geographic Information Systems: A Demonstration, Journal of Environmental Enginering, ASCE, Vol. 122, No. 6.

S. Gholizadeh, A. Pirmoz, and R. Attarnejad., 2011. Assessment of Load Carrying Capacity of Castellated Steel Beams by Neural Networks. Journal of Constructional Steel Research, 67(5):770–779.

S. Haykin., 1999. Neural Networks: A Comprehensive Foundation. Prentice Hall.

Sreedharan, M., Khedr, A. M., & El Bannany, M., 2020. "A Multi-Layer Perceptron Approach to Financial Distress Prediction with Genetic Algorithm." Automatic Control and Computer Sciences 54.6: 475-482.

Shanmuganathan, S., 2016. Artificial neural network modelling: An introduction. Artificial neural network modelling. Springer, Cham, 1-14.

Skalski, J.R., Mackenzie, D.H., 1982. A design for aquatic monitoring programs. J. Environ. Manage. 14, 237–251.

Smith, D.G., McBride, G.B., 1990. New Zealand's national water quality monitoring network—design and first year's operation. Water Resour. Bull. 26, 767–775.

Sivanandam, S. N., and S. N. Deepa., 2008. Genetic algorithms." Introduction to genetic algorithms. Springer, Berlin, Heidelberg, 15-37.

Steuer, R. E., 1986. Multiple Criteria Optimization: Theory, Computation, and Application,New York: John Wiley & Sons.

Spanache, S., Escobet, T., & Travé-Massuyès, L., 2004. Sensor placement optimisation using genetic algorithms. In 15th international Workshop on Principles of Diagnosis (DX'04) (pp. 179-184).

Srinivas, N., & Deb, K., 1994 .Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary computation, 2(3), 221-248.

Shen, C., Wang, L., & Li, Q., 2007. Optimization of injection molding process parameters using a combination of artificial neural network and genetic algorithm method. Journal of materials processing technology 183.2-3: 412-418.

Talbi, E. G., 2009. Metaheuristics: from design to implementation. Vol. 74. John Wiley & Sons.

Tian, Y., Wang, H., Zhang, X., & Jin, Y., 2017. Effectiveness and efficiency of non-dominated sorting for evolutionary multi-and many-objective optimization. Complex & Intelligent Systems, 3(4), 247-263.

Bui, D. T., Bui, K. T. T., Bui, Q. T., Van Doan, C., & Hoang, N. D., 2017. Hybrid intelligent model based on least squares support vector regression and artificial bee colony optimization for time series.

Telci, I. T., Nam, K., Guan, J., & Aral, M. M., 2009. Optimal water quality monitoring network design for river systems. Journal of environmental management 90.10: 2987-2998.

Turabieh, H., Mafarja, M., & Li, X., 2019. "Iterated feature selection algorithms with layered recurrent neural network for software fault prediction." Expert systems with applications 122: 27-42.

Tran, D., H., Cheng, M., Y. and Prayogo, D., 2016. A novel Multiple Objective Symbiotic Organisms Search (MOSOS) for time–cost–labor utilization tradeoff problem Knowledge-Based Systems, 94, 132-145.

Urbieta, M., Oliveira, A., Araújo, J., Rodrigues, A., Moreira, A., Gordillo, S., & Rossi, G., 2014. Web-GIS models: Accomplishing Modularity with Aspects. Innovations System Software Engineering, vol. 10, pp. 59-75.

U.S. Environmental Protection Agency., 2003. Elements of a State Water Monitoring and Assessment Program. Assessment and Watershed Protection Division, Office of Wetlands, Oceans and Watershed. EPA 841-B-03-003.

Yang, X. S., 2011. Metaheuristic optimization. Scholarpedia, 6.8: 11472.

Yang, X., S., 2014. Nature-inspired optimization algorithms, Elsevier.

Yu, X., & Gen, M., 2010. Introduction to evolutionary algorithms. Springer Science & Business Media.

Zdeh, Lotfi A. "Fuzzy sets., 1996. " Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh. 394-432.

Zitzler, E., 1999 .Evolutionary Algorithms for Multiobjective Optimization: Methods and applications (Vol. 63). Ithaca: Shaker.

7. EKLER

Multi-Objective Search

Chromosome

```
using System;
using System.Collections.Generic;
using Catfood.Shapefile;

namespace multiObjectiveSearch
{
	/// <summary>
	/// Description of point.
	/// </summary>
	public class chromosome
	{
		/// <summary>
		/// x and y showing a point, that is the goal we want to optimize
		/// </summary>
		public double x = 0, y = 0;
		/// <summary>
		/// dimentions is number of objectives, number of fittness functions
we should satisfy
		/// </summary>
		private int dimentions = 0;
		/// <summary>
		/// rank[] is an array, indicating output of each fitness function.
		///    length of rank[] array is equal to dimentions.
		///    ranks[i] is value of this answer for fitness functions i.
		/// </summary>
		public double[] rank = null;
		/// <summary>
		/// to integrate all values of rank[] into one value, we use
totalRank.
		///    it shows total value of an answer around all dimentions.
		///    it is used when i want to compare two answers, without using
multi-objective comparing functions.
		///    it could be average of rank[] values.
		/// </summary>
		public double totalRank = 0;
		/// <summary>
		/// settings contaning objective dimentions, region, ...
		/// </summary>
		private RunSettings settings;
		/// <summary>
		/// random generator used for creating new points
		/// </summary>
		private Random pointRandomGenerator = null;
		/// <summary>
		/// indicating which river and which sub-river is containg the point
		/// </summary>
		public double lineIndex = 0, riverIndex = 0;
		/// <summary>
```

```
  /// used for NSGA2 algorithm
  /// </summary>
  public double[] CrowdingDistance = null;
  public double Distance = 0;

  public chromosome(double x_, double y_, ref RunSettings settings_)
  {
         x = x_;
         y = y_;
         dimentions = settings_.dimentions;
         settings = settings_;
         rank = new double[dimentions];
         totalRank = 0;
         pointRandomGenerator = new Random();
         CrowdingDistance = new double[dimentions];
  }
  public chromosome(chromosome c, ref RunSettings settings_)
  {
         x = c.x;
         y = c.y;
         dimentions = settings_.dimentions;
         settings = settings_;
         rank = new double[dimentions];
         totalRank = 0;
         pointRandomGenerator = new Random();
         CrowdingDistance = new double[dimentions];
  }

  public string PrintCMDString()
  {
         string s = "";
         s = s + "x="+ x.ToString() + ",y=" + y.ToString() + ",ranks={";
         for(int i=0; i<dimentions - 1; i++)
                s = s + rank[i].ToString() + ",";
         s = s + rank[dimentions - 1].ToString() + "}";
         return s;
  }

  public string PrintRawStringComplete(string delim)
  {
         string s = "";
         s = s + x.ToString() + delim + y.ToString() + delim;
         for(int i=0; i<dimentions - 1; i++)
                s = s + rank[i].ToString() + delim;
         s = s + rank[dimentions - 1].ToString();
         return s;
  }

  public string PrintRawString(string delim)
  {
//string s = "";
//s = s + x.ToString() + delim + y.ToString() + delim;
//return s;
string s = "";
s = s + x.ToString() + delim + y.ToString() + delim;
for (int i = 0; i < dimentions - 1; i++)
    s = s + rank[i].ToString() + delim;
s = s + rank[dimentions - 1].ToString();
return s;
  }
```

```
            /// <summary>
            /// determining if a point is placed on a line, and if not, put it on
the nearest line.
            /// </summary>
            /// <param name="pop"></param>
            /// <returns>-1 = point is not valid, 0 = point is valid</returns>
            public int ValidatePoint()
            {
                  double mind = double.MaxValue;
                  int mindline = 0, mindshape = 0;
                  //int shapec = 0;

                  //Shapefile sh = settings.GetShape(0);
                  //foreach(Catfood.Shapefile.ShapePolyLine s in sh)
                  for(int i = 0; i < settings.rivers.Count; i++)
                  {
                        //points are on rivers
                        for(int j = 1; j <settings.rivers[i].Count; j++)
                        {
                              double d =
Math.Abs(Geometry.DistanceOfPointToLine(settings.rivers[i][j],

settings.rivers[i][j - 1],

new PointD(x, y)));
                              if(d < mind)
                              {
                                    mind = d;
                                    mindline = j;
                                    mindshape = i;
                                    if(mind == 0)
                                          break;
                              }
                        }
                        //shapec++;
                  }
                  //sh.Dispose();

                  //if mind == 0, it means that this point is placed on one of
lines
                  if(mind == 0)
                        return 0;

                  //fixing point
                  //change position of points, lie it on lines
                  //shapec = 0;
                  //sh = settings.GetShape(0);
                  //foreach(Catfood.Shapefile.ShapePolyLine s in sh)
                  //for(int i = 0; i < settings.rivers.Count; i++)
                  //{
                        //if(shapec < mindshape)
                        //if(i < mindshape)
                        //{
                              //shapec++;
                        //     continue;
                        //}

                        double dx = settings.rivers[mindshape][mindline].X -
settings.rivers[mindshape][mindline - 1].X;
```

```
                            double dy = settings.rivers[mindshape][mindline].Y -
settings.rivers[mindshape][mindline - 1].Y;
                            double b = settings.rivers[mindshape][mindline].Y - (dy
/ dx)*settings.rivers[mindshape][mindline].X;

                            if(dx == 0)
                            {
                                  x = settings.rivers[mindshape][mindline].X;
                                  if(settings.rivers[mindshape][mindline - 1].Y <
settings.rivers[mindshape][mindline].Y)
                                        y =
pointRandomGenerator.NextDouble()*Math.Abs(dy) + settings.rivers[mindshape][mindline
- 1].Y;
                                  else
                                        y =
pointRandomGenerator.NextDouble()*Math.Abs(dy) +
settings.rivers[mindshape][mindline].Y;
                            }
                            else if(dy == 0)
                            {
                                  if(settings.rivers[mindshape][mindline - 1].X <
settings.rivers[mindshape][mindline].X)
                                        x =
pointRandomGenerator.NextDouble()*Math.Abs(dx) + settings.rivers[mindshape][mindline
- 1].X;
                                  else
                                        x =
pointRandomGenerator.NextDouble()*Math.Abs(dx) +
settings.rivers[mindshape][mindline].X;
                                  y = settings.rivers[mindshape][mindline].Y;
                            }
                            else
                            {
                                  if(settings.rivers[mindshape][mindline - 1].X <
settings.rivers[mindshape][mindline].X)
                                        x =
pointRandomGenerator.NextDouble()*Math.Abs(dx) + settings.rivers[mindshape][mindline
- 1].X;
                                  else
                                        x =
pointRandomGenerator.NextDouble()*Math.Abs(dx) +
settings.rivers[mindshape][mindline].X;
                                  y = (dy / dx)*x + b;
                            }

                            double d2 =
Geometry.DistanceOfPointToLine(settings.rivers[mindshape][mindline],

settings.rivers[mindshape][mindline - 1],
                                                                        new PointD(x,
y));
                            if(Math.Abs(d2) > 1E-5)
                            {
                                  Console.WriteLine("error on fixing points");
                                  return -1;
                            }
                            //break;

                            lineIndex = mindline;
                            riverIndex = mindshape;
                      //}
```

```
                    //sh.Dispose();
                    return 0;
                }

                /// <summary>
                /// calculates min distance from current point with important stations
or points of map.
                /// </summary>
                /// <param name="answer"></param>
                /// <returns>min distances from important points or
stations.</returns>
        public void EvaluatePoint()
        {

            //objective 1 : Distance between the river and polluting sources, water
consumption sources, outlet of each sub-basin)

            double mindists = double.MaxValue;
            Shapefile sh = settings.GetShape(1);
            foreach (Catfood.Shapefile.ShapePolygon s in sh)
            {
                double mind = double.MaxValue;
                for (int i = 1; i < s.Parts[0].Length; i++)
                {
                    double d = Math.Abs(Geometry.DistanceOfPointToLine(
                        new PointD(s.Parts[0][i].X, s.Parts[0][i].Y),
                        new PointD(s.Parts[0][i - 1].X, s.Parts[0][i - 1].Y),
                        new PointD(x, y)));
                    if (d < mind)
                        mind = d;
                }
                double d2 = Math.Abs(Geometry.DistanceOfPointToLine(
                        new PointD(s.Parts[0][0].X, s.Parts[0][0].Y),
                        new PointD(s.Parts[0][s.Parts[0].Length - 1].X,
s.Parts[0][s.Parts[0].Length - 1].Y),
                        new PointD(x, y)));
                if (d2 < mind)
                    mind = d2;

                if (mind < mindists)
                    mindists = mind;
            }
            sh.Dispose();

            //objective 2 : distance to roads or telephone lines or cell phone
antennas
            double mindistr = double.MaxValue;
            sh = settings.GetShape(2);
            foreach (Catfood.Shapefile.ShapePolyLine s in sh)
            {
                double mind = double.MaxValue;
                for (int i = 1; i < s.Parts[0].Length; i++)
                {
                    double d = Math.Abs(Geometry.DistanceOfPointToLine(
                        new PointD(s.Parts[0][i].X, s.Parts[0][i].Y),
                        new PointD(s.Parts[0][i - 1].X, s.Parts[0][i - 1].Y),
                        new PointD(x, y)));
                    if (d < mind)
                        mind = d;
                }
```

```
                if (mind < mindistr)
                    mindistr = mind;
            }
            sh.Dispose();

            //return Math.Sqrt(Math.Pow(x-points[objectiveFunctionIndex].x,2)+
            //                 Math.Pow(y-points[objectiveFunctionIndex].y,2));
            //rank = new double[]{mindists, sumdistu, mindistj, mindistp, mindistr};

            rank = new double[] { mindists, mindistr };
            //return new double[]{mindists+ sumdistu+ mindistj+ mindistp+ mindistr};
         }
       }



}
```

Geometry

```
using System;
using Catfood.Shapefile;


namespace multiObjectiveSearch
{
       /// <summary>
       /// Description of Math.
       /// </summary>
       public static class Geometry
       {
              //private double DistanceOfPointToLine(PointD linestart, PointD
lineend, PointD point_)
              public static double DistanceOfPointToLine(PointD linestart, PointD
lineend, PointD point_, bool isSegment = false)
              {
                     /*
                     double d = 0;
                     double dx = linestart.X - lineend.X;
                     double dy = linestart.Y - lineend.Y;
                     d = Math.Abs(dy*point_.X - dx*point_.Y + linestart.X*lineend.Y
- lineend.X*linestart.Y) /
                                   Math.Sqrt(Math.Pow(dx, 2) + Math.Pow(dy, 2));

                     return d;
                     */

                     /*
                     if(isSegment)
                     {
                            double dot1 = dotProduct(linestart, lineend, point_);
                            if(dot1 > 0)
                                   return DistanceOfPointToPoint(lineend, point_);
                            double dot2 = dotProduct(lineend, linestart, point_);
                            if(dot2 > 0)
                                   return DistanceOfPointToPoint(linestart, point_);
                     }

                     return Math.Abs(crossProduct(linestart, lineend, point_) /
DistanceOfPointToPoint(linestart, lineend));
```

```
                    */

                    double mindist = Math.Min(DistanceOfPointToPoint(linestart,
point_), DistanceOfPointToPoint(lineend, point_));

                    bool midx1 = (point_.X <= lineend.X && point_.X >=
linestart.X);
                    bool midx2 = (point_.X >= lineend.X && point_.X <=
linestart.X);
                    bool midy1 = (point_.Y <= lineend.Y && point_.Y >=
linestart.Y);
                    bool midy2 = (point_.Y >= lineend.Y && point_.Y <=
linestart.Y);

                    if(linestart.X == lineend.X)
                    {
                           if(midx1 || midx2)
                                  return point_.X - linestart.X;
                           else
                                  return mindist;
                    }
                    if(linestart.Y == lineend.Y)
                    {
                           if(midy1 || midy2)
                                  return point_.Y - linestart.Y;
                           else
                                  return mindist;
                    }
                    if((midx1 || midx2) && (midy1 || midy2))
                    {
                           double m1 = (linestart.Y - lineend.Y) / (linestart.X -
lineend.X);
                           double m2 = 1 / m1;
                           double b1 = linestart.Y - linestart.X * m1;
                           double b2 = point_.Y - point_.X * m2;
                           double x = (m1 * (b2 - b1)) / (Math.Pow(m1, 2) - 1);
                           double y = m1 * x + b1;
                           //double y2 = m2 * x + b2;

                           if((x >= linestart.X && x <= lineend.X) || (x <=
linestart.X && x >= lineend.X))
                                  return DistanceOfPointToPoint(new PointD(x, y),
point_);
                           return mindist;
                    }
                    else
                    {
                           return mindist;
                    }

             }
             public static double dotProduct(PointD p1, PointD p2, PointD p3)
             {
                    return (p2.X - p1.X) * (p3.X - p2.X) + (p2.Y - p1.Y) * (p3.Y -
p2.Y);
             }
             public static double crossProduct(PointD p1, PointD p2, PointD p3)
             {
                    return (p2.X - p1.X) * (p3.Y - p1.Y) + (p2.Y - p1.Y) * (p3.X -
p1.X);
             }
```

```
		public static double DistanceOfPointToPoint(PointD p1, PointD p2)
		{
			return Math.Sqrt(Math.Pow(p1.X - p2.X, 2) + Math.Pow(p1.Y -
p2.Y, 2));
		}
	}
}
```

NSGA-II

```
using System;
using System.IO;
using System.Collections;
using System.Collections.Generic;

namespace multiObjectiveSearch
{
	public class NSGA2
	{
		public NSGA2(string rs)
		{
			settings = new RunSettings(rs);

			RandomGenerator = new Random();
			pointRandomGenerator = new Random();
			crossoverRandomGenerator = new Random();
			mutationRandomGenerator = new Random();
			mutation_2_RandomGenerator = new Random();
			mutation_3_RandomGenerator = new Random();
			crossover_2_RandomGenerator = new Random();
		}
		private Random RandomGenerator = null, pointRandomGenerator = null,
				mutationRandomGenerator = null
,mutation_2_RandomGenerator = null, mutation_3_RandomGenerator = null,
				crossoverRandomGenerator = null,
crossover_2_RandomGenerator = null;
		private RunSettings settings = null;
		#region NSGA2 functions
		private List<chromosome> FastNondominatedSort(List<chromosome>
population)
		{
			List<chromosome> z = new List<chromosome>();

			validatePopulation(ref population);
			//EvaluateAgainstObjectiveFunctions(population);

			int size = population.Count, M = settings.dimentions;
			int[] n_ = new int[size];
			List<int> [] p_ = new List<int> [size];
			List<List<int>> fronts = new List<List<int>>(0);
			fronts.Add(new List<int>(0));
			int front = 0;

			#region first scan
			for(int i = 0; i < size; i++)
			{
				n_[i] = 0;
				p_[i] = new List<int> (0);
				for(int j = 0; j < size; j++)
				{
```

```
                                if(i == j) continue;

                                int less = 0, more = 0, equal = 0;
                                for(int k = 0; k < M; k++)
                                {
                                        if(population[i].rank[k] >
population[j].rank[k])
                                                less++;
                                        else if(population[i].rank[k] <
population[j].rank[k])
                                                more++;
                                        else equal++;
                                }
                                if(less == 0 && equal != M)
                                        n_[i] ++;
                                else if(more == 0 && equal != M)
                                        p_[i].Add(j);
                        }
                        if(n_[i] == 0)
                        {
                                population[i].totalRank = 1;
                                fronts[front].Add(i);
                        }
                }
                #endregion

                #region find subsequent fronts
                while(fronts[front].Count != 0)
                {
                        List<int> Q = new List<int>(0);
                        for(int i = 0; i < fronts[front].Count; i++)
                        {
                                if(p_[fronts[front][i]].Count > 0)
                                {
                                        for(int j = 0; j <
p_[fronts[front][i]].Count; j++)
                                        {
                                                n_[p_[fronts[front][i]][j]] =
n_[p_[fronts[front][i]][j]] - 1;
                                                if(n_[p_[fronts[front][i]][j]] ==
0)
                                                {

        population[p_[fronts[front][i]][j]].totalRank = front + 1;

        Q.Add(p_[fronts[front][i]][j]);
                                                }
                                        }
                                }
                        }
                        front ++;
                        if(Q.Count == 0)
                                break;
                        fronts.Add(new List<int>(0));
                        fronts[front] = Q;
                }

                #endregion

                #region sorting population based on fronts on Ascented format
                int[] fronts_sorted_index = new int[size];
```

```
                    for(int i = 0; i < size; i++)
                           fronts_sorted_index[i] = i;
                    for(int i = 0; i < size; i++)
                    {
                           for(int j = 1; j < size; j++)
                           {
                                  if(population[j].totalRank > population[j -
1].totalRank)
                                  {
                                         int temp = fronts_sorted_index[j];
                                         fronts_sorted_index[j] =
fronts_sorted_index[j - 1];
                                         fronts_sorted_index[j - 1] = temp;
                                  }
                           }
                    }
                    #endregion

                    #region Crowding Distance
                    int current_index = 0;
                    for(front = 0; front < fronts.Count; front++)
                    {
                           List<chromosome> y = new List<chromosome>(0);
                           for(int i = 0; i < fronts[front].Count; i++)
                                  y.Add(population[fronts_sorted_index[i +
current_index]]);
                           current_index += fronts[front].Count;
                           for(int i = 0; i < settings.dimentions; i++)
                           {
                                  #region sort y based on objective[i]
                                  int[] obj_sorted_index = new int[y.Count];
                                  for(int a = 0; a < y.Count; a++)
                                         obj_sorted_index[a] = a;
                                  for(int a = 0; a < y.Count; a++)
                                  {
                                         for(int b = 1; b < y.Count; b++)
                                         {
                                                if(y[obj_sorted_index[b]].rank[i] <
y[obj_sorted_index[b - 1]].rank[i])
                                                {
                                                       int temp =
obj_sorted_index[b];
                                                       obj_sorted_index[b] =
obj_sorted_index[b - 1];
                                                       obj_sorted_index[b - 1] =
temp;
                                                }
                                         }
                                  }
                                  List<chromosome> y2 = new List<chromosome>(0);
                                  for(int a = 0; a < fronts[front].Count; a ++)
                                         y2.Add(y[obj_sorted_index[a]]);
                                  #endregion

                           double fmax = y2[y2.Count - 1].rank[i];
                                  double fmin = y2[0].rank[i];
                                  y[obj_sorted_index[y.Count -
1]].CrowdingDistance[i] = int.MaxValue;
                                  y[obj_sorted_index[0]].CrowdingDistance[i] =
int.MaxValue;
```

```
                                    for(int j = 1; j < y.Count - 1; j ++)
                                    {
                                          double next_obj = y2[j + 1].rank[i];
                                          double prev_obj = y2[j - 1].rank[i];
                                          if(fmax - fmin == 0)

      y[obj_sorted_index[j]].CrowdingDistance[i] = int.MaxValue;
                                          else

      y[obj_sorted_index[j]].CrowdingDistance[i] = (next_obj - prev_obj) / (fmax -
fmin);
                                    }
                              }

                              for(int i = 0; i < y.Count; i++)
                              {
                                    //double tmp = 0;
                                    double maxd = 0;
                                    for(int j = 0; j < settings.dimentions; j++)
                                    {
                                          //tmp += Math.Pow
(y[i].CrowdingDistance[j], 2);
                                          y[i].Distance += y[i].CrowdingDistance[j];
                                          if(y[i].CrowdingDistance[j] > maxd) maxd =
y[i].CrowdingDistance[j];
                                    }
                                    //y[i].Distance = Math.Pow (tmp, 0.5);
                                    //y[i].Distance = Median(ref
y[i].CrowdingDistance);
                                    //y[i].Distance /= settings.dimentions;
                                    y[i].Distance = maxd;
                              }
                              for(int i = 0; i < y.Count; i++)
                                    z.Add(y[i]);
                        }
                        #endregion

                        return z;
                  }
                  private List<chromosome> TournomentSelection(List<chromosome>
population)
                  {
                        int population_size = population.Count;
                        int poolsize = population_size / 2;
                        int toursize = 2;
                        List<chromosome> pool = new List<chromosome>();
                        int[] selectedmask = new int[population_size];
                        Random r = new Random();

                        for(int i = 0; i < poolsize; i++)
                        {
                              int[] candidate= new int[toursize];
                              double[] objrank = new double[toursize];
                              double[] objdist = new double[toursize];

                              //double minrank = double.MaxValue;
                              double minrank = 0;
                              int minidx = -1;
                              for(int j = 0; j < toursize; j++)
                              {
```

```
                                candidate[j] = Convert.ToInt32 (r.NextDouble() *
population_size - 1);
                                if(candidate[j] == -1) candidate[j] = 0;
                                while(selectedmask[candidate[j]] == 1)
                                        candidate[j] = (candidate[j] + 1) %
population_size;
                                selectedmask[candidate[j]] = 1;

                                objrank[j] = population[candidate[j]].totalRank;
                                objdist[j] = population[candidate[j]].Distance;

                                if(objrank[j] > minrank)
                                {
                                        minrank = objrank[j];
                                        minidx = j;
                                }
                        }

                        //if there is more than 1 obj with min rank, get the 1st
max dist
                        int mincnt = 0;
                        for(int j = 0; j < toursize; j++)
                                if(objrank[j] == minrank)
                                        mincnt++;
                        if(mincnt > 1)
                        {
                                int maxidx = -1;
                                double maxdist = double.MinValue;
                                for(int j = 0; j < toursize; j++)
                                {
                                        if(objrank[j] == minrank && objdist[j] >
maxdist)
                                        {
                                                maxdist = objdist[j];
                                                maxidx = j;
                                        }
                                }
                                if(maxdist == -1)
                                        throw new Exception("این چه وضعشه آخه!!؟؟؟");
                                pool.Add(population[candidate[maxidx]]);
                        }
                        else
                                pool.Add(population[candidate[minidx]]);
                }

                return pool;
        }
        private List<chromosome> CrossowerAndMutation(List<chromosome>
population, int population_size,
                                                double p_crossower, double
p_mutation, int Mutation_MAXDIST)
        {
                //int needed_pop = population_size - population.Count;

                //crate 'population_size' childrens
                List <chromosome> children = new List<chromosome>();
                if(population.Count > 2)
                {
                        for(int i = 0; i < population.Count; i++)
                        {
```

```
                                    double r2 = crossoverRandomGenerator.NextDouble
();
                                    if(r2 > p_crossower)
                                           continue;
                                    int nextp = 0;
                                    do { nextp = crossover_2_RandomGenerator.Next (0,
population.Count); }
                                    while(nextp == i );
                                    chromosome c1 = new chromosome(population[i], ref
settings);
                                    chromosome c2 = new chromosome(population[nextp],
ref settings);

                                    c1.y = population[nextp].y;
                                    c2.y = population[i].y;

                                    children.Add (c1);
                                    if(children.Count >= population_size)
                                           break;
                                    children.Add (c2);
                                    if(children.Count >= population_size)
                                           break;
                             }
                      }
                      //mutation
                      int size2 = children.Count;
                      for(int i = 0; i < size2; i++)
                      {
                             double R = mutationRandomGenerator.NextDouble();
                             if(R < p_mutation)
                             {
                                    chromosome tmp = new chromosome(children[i], ref
settings);
                                    int difdist = mutation_2_RandomGenerator.Next (0,
Mutation_MAXDIST);
                                    double d2 =
mutation_3_RandomGenerator.NextDouble();
                                    if(d2 < 0.25)
                                           tmp.x += difdist;
                                    else if(d2 < 0.5)
                                           tmp.x -= difdist;
                                    if(d2 < 0.75)
                                           tmp.y += difdist;
                                    else
                                           tmp.y -= difdist;
                                    if(tmp != null)
                                           children.Add(tmp);
                             }
                      }

                      return removerepetitiveanswers(children);
                      //return removeWeeklyDivergAnswers(ref childs, benchmarkindex);
              }
              private List<chromosome> Merge(List<chromosome> children,
List<chromosome> population)
              {
                      int childsize = children.Count;
                      int newsize = childsize + population.Count;
                      List<chromosome> union = new List<chromosome>();
                      for(int i = 0; i < childsize; i++)
                             union.Add(children[i]);
```

```
                for(int i = childsize; i < newsize; i++)
                        union.Add(population[i - childsize]);
                return union;
        }
        private List<chromosome> removerepetitiveanswers(List<chromosome>
intermediate_chromosome)
        {
                if(intermediate_chromosome == null ||
intermediate_chromosome.Count == 0)
                        return null;

                int size = intermediate_chromosome.Count;
                List<string> chrstr = new List<string>(0);
                chrstr.Add(intermediate_chromosome[0].PrintRawString(","));
                int reps = 0;
                int []repsa = new int[size];
                for(int i = 1; i < size; i++)
                {
                        string str =
intermediate_chromosome[i].PrintRawString(",");
                        for(int j = 0; j < chrstr.Count; j++)
                        {
                                if(chrstr[j] == str)
                                {
                                        reps++;
                                        repsa[i] = 1;
                                        break;
                                }
                        }
                        if(repsa[i] != 1)
                                chrstr.Add(str);
                }
                //Console.Write(" {0} repetetive answers found.", reps);
                List<chromosome> tmppopnr = new List<chromosome>(0);
                for(int i = 0; i < size; i++)
                        if(repsa[i] != 1)
                                tmppopnr.Add (intermediate_chromosome[i]);
                size -= reps;
                List<chromosome> intermediate_chromosome2 = new
List<chromosome>();
                for(int i = 0; i < size; i++)
                        intermediate_chromosome2.Add(tmppopnr[i]);
                return intermediate_chromosome2;
        }

        private List<chromosome> GenerateExtraPopulation(int PopulationSize)
        {
                List<chromosome> population = new List<chromosome>();

                for(int i = 0; i < PopulationSize; i++)
                {
                        double x =
pointRandomGenerator.Next(Convert.ToInt32(Math.Round(settings.region.Left)),

Convert.ToInt32(Math.Round(settings.region.Right)));
                        double y =
pointRandomGenerator.Next(Convert.ToInt32(Math.Round(settings.region.Top)),

Convert.ToInt32(Math.Round(settings.region.Bottom)));
                        population.Add(new chromosome(x, y, ref settings));
                }
```

```
                return population;
            }
            private void validatePopulation(ref List<chromosome> pop)
            {
                for(int i = 0; i < pop.Count; i++)
                {
                    if(i > pop.Count - 1)
                        return;
                    int d = pop[i].ValidatePoint();
                    if(d < 0)
                    {
                        pop.RemoveAt(i);
                        i--;
                        continue;
                    }
                    pop[i].EvaluatePoint();
                }
            }
            #endregion

            public List<chromosome> SearchDesignSpace()
            {
                #region reading settings
                int t = 0;
                int Maxt = settings.ReadValueOfInteger(settings.RootAddress,
"NSGA2_InnerLoopMaxRuns");
                int PopulationSize =
settings.ReadValueOfInteger(settings.RootAddress, "NSGA2_population_size");
                double P_Crossover =
settings.ReadValueOfDouble(settings.RootAddress, "NSGA2_P_Crossover");
                double P_Mutation =
settings.ReadValueOfDouble(settings.RootAddress, "NSGA2_P_Mutation");
                int MAX_NSGA2_Runs = settings.ReadValueOfInteger
(settings.RootAddress, "NSGA2_OuterLoopMaxRuns") - 1;
                bool initial_import_type =
settings.ReadValueOfString(settings.RootAddress, "initial_import_type")
                    == "add";
                #endregion

                List<chromosome> pop_collection = new List<chromosome>();
                for(int NSGA2_Counter = 0; NSGA2_Counter  < MAX_NSGA2_Runs;
NSGA2_Counter ++)
                {
                    Console.WriteLine("\n\nrunning NSGA2 for " +
(NSGA2_Counter  + 1).ToString() + "th times :");

                    //initialize population
                    #region main loop
                    //List<chromosome> parents =
GenerateExtraPopulation(PopulationSize);
                    List<chromosome> parents =
settings.ReadInitialPoints(settings.RootAddress, "initial_points", ref settings);
                    if(parents == null)

        parents.AddRange(GenerateExtraPopulation(PopulationSize));
                    else if(initial_import_type)

        parents.AddRange(GenerateExtraPopulation(PopulationSize - parents.Count));
                    validatePopulation(ref parents);
                    // running for a fixed iterations
```

```
                        t = 0;
                        while(t < Maxt)
                        {
                                if(parents == null || parents.Count == 0)
                                        break;
                                List<chromosome>selected =
FastNondominatedSort(parents);
                                if(selected == null || selected.Count == 0)
                                        break;
                                selected = TournomentSelection(selected);
                                if(selected == null || selected.Count == 0)
                                        break;
                    t++;
                                List<chromosome> ltmp =
FastNondominatedSort(selected);

       Console.WriteLine(Convert.ToInt32(Math.Round(ltmp[0].rank[0])).ToString()+
                        ","+

Convert.ToInt32(Math.Round(ltmp[0].rank[1])).ToString());

                                List<chromosome> children =
CrossowerAndMutation(selected, PopulationSize / 2, P_Crossover, P_Mutation,
3000000);
                                List<chromosome> randomPop = new
List<chromosome>();
                                if(children == null || children.Count == 0)
                                        randomPop =
GenerateExtraPopulation(PopulationSize - selected.Count);
                                else
                                        randomPop =
GenerateExtraPopulation(PopulationSize - selected.Count - children.Count);
                                //List<chromosome> randomPop =
GenerateExtraPopulation(children.Count / 2);
                                List<chromosome> union = new List<chromosome>();
                                if(randomPop == null || randomPop.Count == 0)
                                        continue;
                                else
                                        union = Merge(Merge(children, selected),
randomPop);
                                parents = removerepetitiveanswers(union);
                                if(union == null || union.Count == 0)
                                        break;
                                validatePopulation(ref parents);


                        }
                        #endregion
                        pop_collection.AddRange(parents);
                        pop_collection =
removerepetitiveanswers(pop_collection);
                    }

                    //select best answer from all iteration
                    return
TournomentSelection(FastNondominatedSort(pop_collection));

                    //return pop_collection;
            }
       }
```

```
}
```

Program

```
using System;
using System.Collections.Generic;
using System.IO;


namespace multiObjectiveSearch
{
       class Program
       {

              public static void Main(string[] args)
              {

                     NSGA2 n = new NSGA2("settings.txt");

                     List<chromosome> ansn = n.SearchDesignSpace();

                     StreamWriter sw = null;
            StreamWriter sw1 = null;

                     try
                     {
                            sw = new StreamWriter("NSGA2_output1.txt");

                     }
                     catch
                     {
                            if(sw != null)
                                   sw.Close();
                     }
            if (sw != null)
            {
                sw.WriteLine(String.Format("{0}\t{1}\t{2}\t{3}", "lat", "lon", "
f1", "f2"));
                for (int i = 0; i < ansn.Count; i++)
                {
                    //////////////////////////
                    double c14 = 6378137, c15 = 6356752.31424518;
                    double x = ansn[i].x, y = ansn[i].y;
                    char c24 = 'N';
                    double c17, c18, c19, c20, c23, o6, f6, p6, e6, q6, k6, l6, r6,
s6, t6, u6, v6, w6, x6, y6, z6, aa6, ab6, ac6, ad6, ae6, af6, l1, l2, m6, n6;
                    c17 = Math.Sqrt(Math.Pow(c14, 2) - Math.Pow(c15, 2)) / c14;
                    c18 = Math.Sqrt(Math.Pow(c14, 2) - Math.Pow(c15, 2)) / c15;
                    c19 = Math.Pow(c18, 2);
                    c20 = Math.Pow(c14, 2) / c15;
                    c23 = 37;
                    f6 = y;
                    e6 = x;
                    if (c24 == 'S')
                        o6 = f6 - 10000000;
                    else
                        o6 = f6;
                    k6 = o6 / (6366197.724 * 0.9996);
```

```
                    l6 = (c20 / Math.Pow((1 + c19 * (Math.Pow(Math.Cos(k6), 2))),
(0.5))) * 0.9996;
                    p6 = (e6 - 500000) / l6;
                    q6 = Math.Sin(2 * k6);
                    r6 = q6 * (Math.Pow(Math.Cos(k6), 2));
                    s6 = k6 + (q6 / 2);
                    t6 = (3.00 * s6 + r6) / 4;
                    u6 = (5 * t6 + r6 * (Math.Pow(Math.Cos(k6), 2))) / 3;
                    v6 = (3.00 / 4.00) * c19;
                    w6 = (5.00 / 3.00) * Math.Pow(v6, 2);
                    x6 = (35.00 / 27.00) * Math.Pow(v6, 3);
                    y6 = 0.9996 * c20 * (k6 - (v6 * s6) + (w6 * t6) - (x6 * u6));
                    z6 = (o6 - y6) / l6;
                    aa6 = ((c19 * Math.Pow(p6, 2)) / 2) * Math.Pow(Math.Cos(k6), 2);
                    ab6 = p6 * (1 - (aa6 / 3));
                    ac6 = z6 * (1 - aa6) + k6;
                    ad6 = (Math.Exp(ab6) - Math.Exp(-ab6)) / 2;
                    ae6 = Math.Atan(ad6 / Math.Cos(ac6));
                    af6 = Math.Atan(Math.Cos(ae6) * Math.Tan(ac6));
                    m6 = k6 + (1 + c19 * (Math.Pow(Math.Cos(k6), 2) - (3.00 / 2.00)
* c19 * Math.Sin(k6) * Math.Cos(k6) * (af6 - k6))) * (af6 - k6);
                    n6 = 6 * c23 - 183;
                    ansn[i].x = (m6 / Math.PI) * 180;
                    ansn[i].y = (ae6 / Math.PI) * 180 + n6;

  // sw.WriteLine(String.Format("{0}\t{1}\t{2}\t{3}\t{4}\t{5}\t{6}", l1.ToString(),
l2.ToString(), " title", "description", "25,25", "0,0",
"/Content/img/location_iconResize.png"));
                    //sw.WriteLine(ansn[i].PrintRawString("\t"));
                }
            }
                     sw.Close();
            try
            {
                sw1 = new StreamWriter("NSGA2_output2.txt");
            }
            catch
            {
                if (sw1 != null)
                    sw1.Close();
            }
            if (sw1 != null)
            {
                sw1.WriteLine(String.Format("{0}\t{1}\t{2}\t{3}\t{4}\t{5}\t{6}",
"lat", "lon", " title", "description", "iconSize", "iconOffset", "icon"));
                for (int i = 0; i < ansn.Count; i++)
                {

                    sw1.WriteLine(String.Format("{0}\t{1}\t{2}\t{3}\t{4}\t{5}\t{6}",
ansn[i].x.ToString(), ansn[i].y.ToString(), " title", "description", "25,25", "0,0",
"/Content/img/location_iconResize.png"));
                    //sw.WriteLine(ansn[i].PrintRawString("\t"));
                }
            }
            sw1.Close();
              }

       }
}
```

ÖZGEÇMİŞ

2003-2007 yılları arasında Urmia üniversitesinde lisans eğitimini tamamladı. 2009-2011 yılları arasında İslami Azad Üniversitesi, Bilim ve Araştırma Şubesi, Tahran da yüksek lisans eğitimini tamamladı. 2010-2011 yılları arasında İran Ulusal Coğrafya Teşkilatı'da araştırmacı olarak çalıştı. ve 2011-2014 yılları arasında Urmiye Payame Noor Üniversitesi ve Urmiye Uygulamalı Bilim ve Teknoloji Üniversitesi'nde öğretim görevlisi olarak çalıştı. 2014 yılında Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Harita Mühendisliği Anabilim Dalı'nda doktora eğitimine başladı. 2020 yılında Hekmat Üniversitesi Harita Mühendisliği Bölümünde öğretim görevlisi olarak göreve başladı ve halen burada göreve devam etmektedir. 2019 yılında 10 ay Wroclaw'da doktora araştırmaları yaptı. İyi derecede İngilizce bilmektedir.