NP PROBLEMLERDE VE SİSTEM TASARIMINDA DNA HESAPLAMANIN KULLANILMASI

Uğur ÇİĞDEM

Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Yrd. Doç. Dr. Mehmet KARAKÖSE
EKİM-2011

T.C
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

NP PROBLEMLERDE VE SİSTEM TASARIMINDA DNA HESAPLAMANIN KULLANILMASI

YÜKSEK LİSANS TEZİ

Uğur ÇİĞDEM

Tezin Enstitüye Verildiği Tarih: 30.09.2011
Tezin Savunulduğu Tarih: 26.10.2011

Tez Danışmanı: **Yrd. Doç. Dr. Mehmet KARAKÖSE (F.Ü)**

Diğer Jüri Üyeleri: **Prof. Dr. Yakup DEMİR (F.Ü)**

Prof. Dr. Erhan AKIN (F.Ü)

EKİM-2011

ÖNSÖZ

Bu tezde, NP problemlerin optimizasyonu ve denetleyici parametrelerinin ayarlanması için sayısal ortamda çevrimiçi ve çevrimdışı DNA hesaplama algoritması geliştirilmiştir. Çalışmanın devamında DNA hesaplama parametreleri QPSO algoritması ile uyarlanabilir duruma getirilmiştir. Algoritmanın çözelti ortamındaki performansını göstermesi için DNA bilgisayarlara ihtiyaç duyulmaktadır. DNA hesaplama için kullanılan bu bilgisayarların çok kısa bir zamanda içerisinde geliştirilerek pratik kullanımının sağlanmasını umuyorum. DNA bilgisayarların kullanıcılara ulaşmasıyla DNA hesaplama alanında hızlı bir ilerleme sağlanacak olup, birçok optimizasyon problemi bugünkünden daha hızlı çözülecek ve daha net sonuçlar elde edilecektir. Bu konudaki arzumuz DNA bilgisayarların hızla geliştirilerek en kısa zamanda bizlerin kullanımına sunulmasıdır.

Bu tez çalışmamda değerli vaktini bana harcayarak ve gerekli ortamı sağlayarak çalışmamın bitirilmesinde her türlü desteği veren sayın hocam Yrd. Doç. Dr. Mehmet KARAKÖSE'ye çok teşekkür erdim. Tez çalışmam esnasında değerli vaktini ayırarak bana yardımcı olan sayın Dr. İlhan Aydın'a teşekkür ederim. Ayrıca tez çalışmam boyunca bana destek olan, maddi ve manevi sıkıntılara katlanan sevgili eşime çok teşekkür ederim.

Uğur ÇİĞDEM
ELAZIĞ - 2011

İÇİNDEKİLER

Sayfa No

ÖZET

Günümüze kadar birçok problem, matematiksel denklem ve modeller kullanılarak çözülmüştür. Fakat, karmaşık ve zor problemlerin optimizasyonu geleneksel yöntemlerle çözülemeyebilir. Özellikle, problemin parametre sayısı ve çözümün zorluk derecesi arttıkça algoritmanın tamamlanması daha çok zaman alır. Bu nedenle, yapılan bilimsel çalışmalarda bu tür problemlerin çözümü için optimizasyon algoritmaları geliştirilmiştir. Optimizasyon algoritmaları, belirli bir çözüm alanında rastlantısal olarak arama yapar ve matematiksel denklemlere gerek duymaz. Son yıllarda DeoksiriboNükleikAsit (DNA) hesaplama adı verilen bir optimizasyon yöntemi araştırmacılar tarafında farklı polinomal olmayan (Non Polinomial-NP) problemlere uygulanmıştır. Bu algoritma, DNA moleküllerini kodlayarak optimizasyon problemlerini çözer. DNA molekülü, veri saklama kapasitesi ve hızlı işlem yapma yeteneği ile literatürde yeni bir hesaplama alanı oluşturmaktadır.

Bu çalışma sırasında NP problemlerin optimizasyonu ve denetleyici parametrelerinin ayarlanması için DNA hesaplama algoritması geliştirilmiştir. Doğal DNA hesaplama algoritması çözelti ortamında uygulanırken, bu çalışmada sayısal biçime dönüştürülerek optimizasyon problemlerine uygulanmıştır. Ayrıca sayısal DNA hesaplama algoritması; gezgin satıcı, sırt çantası problemi ile PI ve bulanık denetleyici parametrelerinin ayarlanması gibi üç farklı probleme uygulanmıştır. DNA hesaplamanın benzetim sonuçları diğer optimizasyon yöntemleri ile karşılaştırılmış, daha performanslı değerler elde edilmiştir.

Anahtar Kelimeler: DNA Hesaplama, Gezgin Satıcı Problemi, Sırt Çantası Problemi, PI ve Bulanık Denetleyiciler

SUMMARY

Using of DNA Computing Algorithm for NP Problems and System Design

Most of the problems have been solved by using mathematical equations and models up to now. But the optimization of complex and hard problems are may not be solved by using traditional methods. Especially when the number of parameters and the degree of hardness increase, the solution of these problems takes more time.Therefore, optimization algorithms have been developed to solve this kind of problems.Optimization algorithms randomly search the solution of a problem in a determined space and they needn't mathematical equations. In recent years, an optimization algorithm named DNA computing is applied to different NP problems by researchers. This algorithm solves optimization problems by encoding DNA molecules. DNA molecule constitutes a new computation area in literature with its large data storage and parallel processing capability. In this study, DNA computing algorithm is used for optimization of NP problems and to tuning PI and fuzzy parameters.While natural DNA computing algorithm is implemented on solution environment, it is applied to optimization problems by transforming to a numeric form in this study.In this thesis, numeric DNA computing algorithm is applied tothree different problems such as travelling salesman, knapsack, and tuning of PI and fuzzy parameters. The simulation results of the DNA computing are compared to other optimization methods and good results are obtained.

Key Words: DNA computing, travelling salesman problem, knapsack problem, PI controller, Fuzzy controller

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

SEMBOLLER LİSTESİ

A : Adenin

C : Sitozin

G : Guanin

H : Yükseklik

K : Kırmızı

K1 : Kazanç-1

K2 : Kazanç-2

K3 : Kazanç-3

Kd : Türev Sabiti

Ki : İntegral Sabiti

Kp : Oransal Sabiti

M : Mavi

P : Polinomial

R : Çap

T : Timin

Tm : Erime sıcaklığı

V : Hacim

Y : Yeşil

Π : Sabit

KISALTMALAR

BTM : Bireysel Topluluk Modeli

DNA : Deoksiribonükleikasit

EA : Evrimsel Algoritmalar

EMCC : Europe Moleculer Computing Committee

GA : Genetik Algoritma

IAE : Integral Absolute Error

IP : İnternet Protokol

ISE : Integral Square Error

ITAE : Integral Time Absolute Error

ITSE : Integral Time Square Error

NP : Non Polinomial

PCR : Polimeraz Chain Reaction

PI : Proportional Integral

RNA : Ribonükleikasit

SAT : Satisfiability

SA : Self Assembly

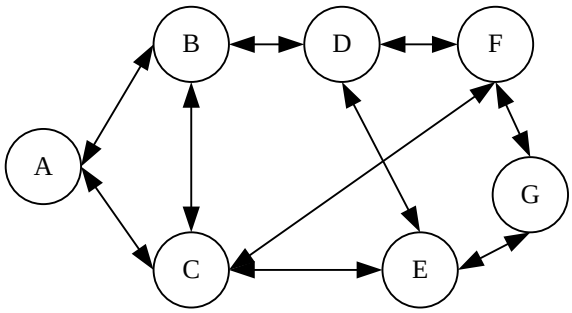
Tm : Melting Temperature

QPSO : Kuantum Parçacık Sürü Optimizasyonu

YBS : Yapay Bağışık Sistemler

1. GİRİŞ

DNA hesaplama, DNA moleküllerini kullanarak hesaplama yapan yeni bir araştırma alanı olarak hızla büyümekte ve gelişmektedir. Bu durum DNA moleküllerinin eşsiz özelliklerinin kullanımı ile gerçekleşmektedir. Çünkü DNA molekülleri paralel işlem yapma, hızla çoğalma ve çok büyük miktarda veri saklama kapasitesine sahiptir. DNA hesaplama alanında bugüne kadar birçok bilimsel çalışma gerçekleştirilmiştir. Bu çalışmalardan ilki Feynman tarafından yapılmıştır [1]. Feynman, DNA moleküllerini kullanarak bilgisayar elde etmek için çalışmıştır. Teorik alandaki bu çalışmaların, uygulamada başarılması uzun süre mümkün olmamıştır. DNA molekülleri kullanılarak yapılan ilk uygulama Adleman tarafından biyoloji laboratuarında gerçekleştirilmiştir [2]. Bu uygulamada 7 şehirden oluşan bir gezgin satıcı problemi DNA dizileri kullanılarak kodlanmış ve problemin çözümü elde edilmiştir. Şekil 1.1'de bu uygulamaya benzer örnek bir gezgin satıcı problemi görülmektedir.

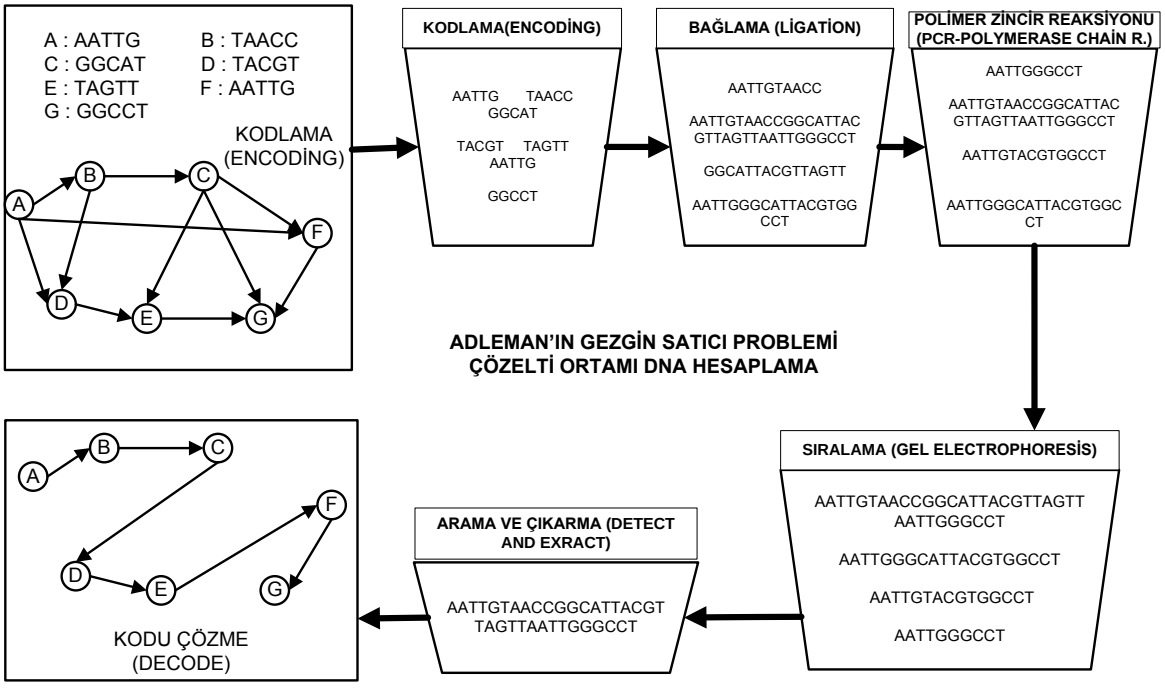


Şekil 1.1. Örnek bir gezgin satıcı problemi

Sayısal bilgisayarlarda veriler ikili sayı sistemi ile kodlanırken, Adleman çalışmasında problemin kodlanması için DNA moleküllerini, hesaplama yaparken de DNA moleküllerine ait ayrılma, birleşme, hibritleşme ve polimerleşme gibi özelliklerden yararlanmıştır. Problem DNA ile kodlanırken, her bir düğüm ve bu düğümleri birleştiren bağlantılar için 20 nükleotit kullanılmıştır. Düğümler ile bağlantılarını gösteren nükleotit dizileri, tek bir tüpe konularak çözüm olabilecek bütün yollar Polimer zincir reaksiyonu (PCR-Polymerase Chain Reaction) yöntemi ile oluşturulmuştur. Bu karışımda, düğümler ve bağlantılar birbirlerinin tümleyeni oldukları için kolayca birleştirilmiş ve uzun DNA

dizileri elde edilmiştir. Deney sonuçları gel electrophoresis metodu ile elektroskobik floresan üzerine yazdırılmıştır. Bu problemin amacı, sabit bir düğümden başlayıp tüm düğümlere birer kez uğradıktan sonra aynı noktaya en kısa yoldan gelmektir. Şekil 1.2'de bu adımların uygulanışı gösterilmektedir. İlk kez kullanılan bu yöntem, başarılı bir şekilde uygulanmış olup DNA dizileri ile mümkün olan bütün çözümler elde edilmiştir. Bu deneyle DNA dizilerinin hesaplama yeteneğini ortaya koymaya çalışmıştır. Adleman bu çalışmalarını "Kombinasyon problemlerinin moleküler hesaplama ile çözümü" adı ile seminer çalışması olarak yayımlamıştır [2]. Daha sonra birçok araştırmacı bu deneyden esinlenerek karmaşık NP problemlerin DNA dizileri ile çözümünü gerçekleştirmiştir [3-13].

DNA hesaplamayı laboratuar ortamında ilk olarak Adleman kullanmıştır. Onun bu çalışması ile moleküler hesaplama alanında yeni bir yöntem literatüre geçmiştir. Adleman'ın gezgin satıcı probleminin çözümü için çözelti ortamında kullandığı adımlar Şekil 1.2'de verilmektedir.



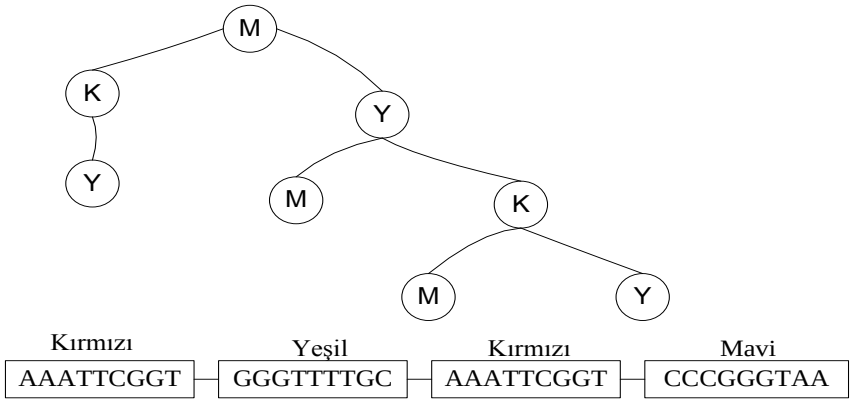
Şekil 1.2. Gezgin satıcı probleminin çözelti ortamında çözüm aşamaları

DNA dizilerinin biyolojik ortamda gerçekleştirdiği birçok işlem, DNA hesaplamanın temelini oluşturmaktadır. Bu işlemlerin kullanımı ile önceleri çok uzun zaman alan birçok

problem kısa sürede çözülmektedir. Özellikle Adleman'ın çalışmasını takip eden yıllarda, DNA hesaplama çok yönlü olarak hızla gelişmiş ve heyecan veren bir noktaya ulaşmıştır. Bu alandaki çalışmalar Lipton ile devam etmektedir. Lipton mantıksal denklemlerden oluşan SAT (Satisfiability) problemlerinin DNA hesaplama ile çözümü üzerine bir çalışma yapmıştır [14, 15]. Yaptığı çalışmada, bütün çözüm kümesini DNA dizileri ile inşa etmiş ve bu dizileri 1 ve O veya True ve False değerleri ile göstermiştir. Bu çalışmada sentez, arama, polimerleşme ve ayırma gibi biyolojik adımları izleyerek laboratuar ortamında milyonlarca biyokimyasal tepkime gerçekleştirmiştir. DNA dizileri ile muhtemel bütün çözüm kümesini kodlayıp çoğaltarak, aynı özellikli test tüplerine yerleştirip paralel işlem uygulamış ve çözümü elde etmiştir. Ouyang ve arkadaşları, DNA moleküllerini kullanarak maksimum klikleme problemlerinin çözümü için yeni bir yöntem geliştirmiştir [16]. Çalışmada 7 düğümlü ve her düğümün kenarlarla birbirine bağlandığı grafik problemini örnek almıştır. Burada maksimum düğümden oluşan kümeyi elde etmeye çalışmıştır. Her bir düğümü 6 bitlik ikili sayı sistemi ile kodlamıştır. Seçilen örnekte düğüm sayısı az olduğu için çok fazla sorun çıkmamakla beraber düğüm sayısı artırıldığında problemin çözümünün zorlaştığı görülmüştür. Tomohiro ve arkadaşları, DNA molekülleri ile kodlamaya dönük yeni bir yöntem önermiş ve çalışmalarında uygulamıştır [17, 18]. Maley, bilgisayar programlama ve kimyasal tepkimeler ışığında DNA hesaplamanın kullanımına yönelik bir çalışma gerçekleştirmiştir [19]. Wood ve arkadaşları, Max 1 problemlerin çözümünde DNA hesaplama algoritmasının genetik algoritmadan daha iyi olduğunu ileri sürmüş ve çalışmalar yapmıştır [20]. Chen ve arkadaşları, şu anki moleküler biyoloji tekniklerinin DNA hesaplama tekniklerini tamamlamak için kullanılacağını gösteren bir çalışma gerçekleştirmiştir [21]. Ding ve Ren'e göre, DNA hesaplama ile ayarlanabilir Takagi-Sugeno bulanık kuralların bulunması mümkün olmaktadır [22]. Yanfeng ve arkadaşları DNA bilgisayarlar ile elektronik bilgisayarların karşılaştırmasını yapan bir çalışma gerçekleştirmiş ve DNA bilgisayarların daha avantajlı olduğunu savunmuştur [23].

Qiu ve Lu, DNA hesaplamanın 3-renk problemlerine uygulamasını gerçekleştirmişlerdir [24, 25]. 3-renk uygulamalarda bir problemin çözümü için doğrusal olmayan grafikler kullanılmaktadır. Her problem, grafikler kullanılarak alt birimlere ayrılmakta ve 3 renk kullanarak kodlanmaktadır. Sonra paralel işlem kullanılarak alt birimler birleştirilmekte ve renk havuzları oluşturulmaktadır. İşlemin sonunda, renk havuzlarından hatasız olan havuz problemin çözümünü vermektedir. Grafiklerin renklendirilmesi problemlerinde her bir

çözüm veya değişken, renk gruplarından biri seçilerek gösterilir. Her bir komşu çözüm veya durum aynı renk ile gösterilemez. 3-Renk ile kodlamada, küçük ebatlı problemler için fazla sorun olmamakla birlikte, çok büyük kapasiteli problemlerde zorluk olmaktadır. Bu problemlerde öncelikle mümkün olan bütün çözüm yollarının renk haritasının çıkarılması gerekir. Burada düşünülen her bir çözüm tekli DNA dizileri kullanılarak gösterilir. Bu gösterimler için birçok DNA dizisi birleştirilerek büyük DNA dizileri oluşturulur. Eğer problemlerin çözümü için 3 renk kullanılacaksa kırmızı, yeşil ve mavi seçilir.



Şekil 1.3. Örnek bir problemin 3-renk kullanılarak kodlanması

3 renk kodlama yöntemi, NP problemlerin çözümünde büyük kolaylıklar sağlamaktadır. Şekil 1.3'te örnek bir problemin 3-renk yöntemi kullanılarak kodlanması gösterilmektedir. Kodlama yapılırken, her bir renk belirli uzunlukta DNA dizileri ile kodlanmaktadır. DNA dizilerinin uzunluğu uygulayıcı tarafından belirlenmekte ve problemin boyutu dikkate alınarak karar verilmektedir. Burada kullanılan her bir renk, problemin çözüm elemanlarını göstermektedir.

Forbes, dijital bilgisayarlar ile DNA bilgisayarları karşılaştıran bir çalışma yaparak DNA hesaplamanın paralel işlem yapma özelliği olduğunu, bu özellik sayesinde dijital bilgisayarlardan milyon kez daha etkili olabildiğini belirtmiştir [26]. Tsuboi ve arkadaşları, DNA dizileri kullanıldığında 200 petabit veri saklama kapasitesine sahip depolama aygıtları yapılabileceğini ileri sürmüşlerdir [27]. Boneh ve arkadaşları, DNA molekülleri ile bilgi güvenliği ve verilerin şifrelenmesi üzerine bir çalışma yapmıştır [28]. Clelland ve arkadaşları, güvenlik ile ilgili mesajları DNA dizileri ile şifreleyerek DNA moleküllerinin içerisine gizlemiş böylece DNA moleküllerinin kolayca çoğaltılarak iyi bir veri saklama

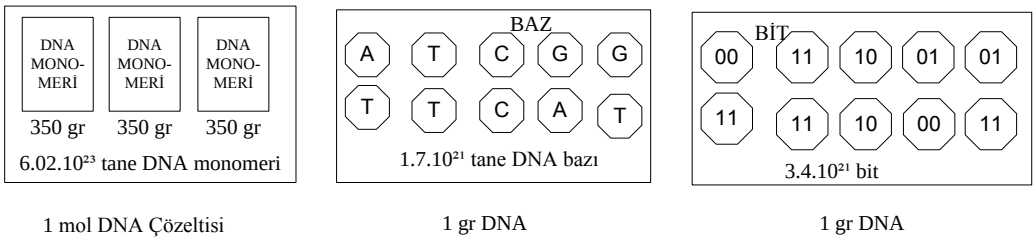
alanı olarak kullanılabileceğini göstermiştir [29, 30]. Milyonlarca DNA dizisi içerisinde gizlenen verinin bulunması mümkün değildir. Veri şifrelerinin çözümü ancak şifreleme algoritmasını bilen kişiler tarafından mümkündür. Shin ve arkadaşları, DNA molekülleri arasındaki hidrojen bağları sayısını kullanarak yeni bir ağırlık kodlama metodu geliştirip yaptığı çalışmada, DNA dizilerini gerçek değerler ile göstererek moleküler programlama gerçekleştirmiştir [31]. Kodlama işlemi sabit kod uzunluğu ve kenarların ağırlıkları dikkate alınarak yapılmıştır. Şekil 1.4'te DNA molekülleri ile örnek bir şifreleme gösterilmektedir. Herhangi bir veriyi simgeleyen TTTC dizisi binlerce DNA dizisinin içerisine belirlenen bir algoritma ile gizlenmektedir.

AATC	TCCG	TCAA	AATC	CCCT
TTCA	GCCC	TCCC	GACA	AATC
TTTG	ATCA	GGGA	TCAG	CAAT
GGCC	AATC	TTTC	GCAA	GGGA
TCGA	TAAA	AATC	GGGG	TTTT
GGGA	GCCC	CCCA	TTTC	CCTA

Şekil 1.4. Verilerin DNA ile şifrelenmesi

DNA yapıları ve fonksiyonları üzerine yapılan çoğu çalışmalar neticesinde, hesaplama alanında belirli bir amaca ulaşılmıştır [32, 33]. Bu amaçlar şu şekilde sıralanabilir: Uzun DNA dizileri, kodlamaları büyük oranda kolaylaştırmış olup aynı zamanda bu kodlamalar birçok çalışmada ikili sayı sistemlerine çevrilmiştir. Dört nükleikasit kodlanırken dört olasılık olmasına rağmen bu ikili sayı sistemlerine çevrilirken 0 ve 1 rakamları kullanılarak kodlanmıştır. Burada DNA dizilerinde kullanılan dört bazdan ikisi diğer ikisinin tümleyeni olduğu için kolayca ikili sistemde kullanılabilmektedir. Birçok problemde çözüm kümeleri ve değişkenler bu şekilde DNA dizileri ile saklanmaktadır. Baum, DNA dizilerini kullanarak veritabanı oluşturmuşlardır [34]. Düşünce şekli optimizasyon problemleri ile aynı olan veritabanı işleminde, DNA dizileri veritabanına girilen veriler olarak düşünülmüştür. DNA hesaplamada kullanılan bütün işlemlerin, burada da kullanılması amaçlanmıştır. DNA dizileri sadece karşılaştırma işleminde olmayıp veri girişi gibi birçok noktada mükemmel bir veritabanı bileşeni oluşturmuştur. Reif ve arkadaşları tarafından yapılan bu çalışmada 1 mol çözeltinin 6.02×10^{23} tane DNA monomeri içerdiği belirlenmiştir [35]. Bir DNA monomeri yaklaşık olarak 350 gram/mol ağırlığına sahiptir. 1

gram DNA $1.7x10^{21}$ DNA bazı içerir. 4 DNA bazı 2 bit ile kodlandığından 1 gram DNA $3.4x10^{21}$ bit veri depolama yeteneğine sahiptir. Günümüzde kullanılan bilgisayarlar, 1 gram için 10^9 bit veri saklama kapasitesine sahiptir. Bu durumda, DNA moleküllerinin diğer depolama birimlerinden 10^{12} kat daha fazla veri saklama kapasitesine sahip olduğu açıkça görülmektedir. Bu bilgilerden görüldüğü üzere, birkaç gram DNA kullanılarak milyonlarca kütüphane kapasitesindeki veri saklanabilmektedir. Şekil 1.5'te DNA moleküllerinin veri saklama kapasiteleri şematik olarak gösterilmektedir. Çalışmada ayrıca DNA hesaplamanın, DNA moleküllerinden oluşan çok yüksek kapasiteli veri tabanlarında hızlı veri arama için kullanılabileceğini belirtmiştir. Yapılan değerlendirmede, DNA moleküllerinin paralel işlem yapma yeteneğine vurgu yapılmış ve bu alana ilginin artması sağlanmış olup, paralel işlem yapma sayesinde problemlerin çok kısa zamanda çözülebileceği öngörülmüştür.



Şekil 1.5. DNA molekülünün veri saklama kapasitesinin gösterimi

J. Y. Lee ve arkadaşları, DNA moleküllerinin erime sıcaklığını (Tm) kullanarak gezgin satıcı probleminin kodlanması ve çözümü üzerine bir çalışma gerçekleştirmiştir [36]. Bu çalışma moleküllerin sıcaklıklarının değişkenlik göstermesi, A ile T bazlarının G ile C bazlarından daha düşük sıcaklıkta ayrılma ve birleşme gerçekleştirmesi nedeniyle istenilen düzeyde başarıya ulaşmamıştır.

DNA kullanılarak yapılacak hesaplamalar, genellikle laboratuar ortamlarında inşa edilir. Teknik olarak AAAGTTCC gibi DNA dizilerini bilgisayar ortamında oluşturmak çok zor değildir ve kısa bir bekleme süresinden sonra üretilir. Fakat buradaki problem bu dizilerin DNA hesaplamada nasıl kullanılacağı ve DNA moleküllerinin çözelti ortamında gerçekleştirdiği birçok işlemin elektronik ortama uyarlanması hususudur.

DNA hesaplama ile birlikte yeni deneyler, algoritmalar ve teorik prensipler geliştirilmiştir. Splicing ve Membrane sistemler DNA hesaplamanın kullanıldığı önemli

çalışmalardan biridir. Bu alanda Head ve arkadaşları bir çalışma yaparak DNA dizilerinin biyolojik ortamlarda kullandığı Ligase, mutasyon ve crossover gibi metotlardan yararlanılmıştır [37]. Splicing sistemler ile hesaplama kapasitesi artırılarak hesaplama fonksiyonları yazılmıştır. Yapılan bu çalışmada laboratuar ortamında DNA hesaplama ile grafiği oluşturan düğüm kümesi içerisinden maksimum bağımsız alt küme sayısının bulunması amaçlanmıştır. Düğümleri DNA dizi segmentleri ile kodlamış enzimler kullanarak çözüm kümesini oluşturan DNA dizi parçaları, çözelti ortamında ayrılmıştır. Bu tür çalışmalarda, düğüm sayısı arttığında enzim sayısındaki sınırlamadan dolayı çözüm kümesinin bulunması zor olmaktadır. Membrane sistemler ise veri yapılarını modellemek için kullanılmıştır.

DNA hesaplama, bilgisayar ağlarının yerleştirilmesinde ve tasarımında da kullanılmaktadır. Ağlarda kullanılan yönlendiricilerin yerleştirilmesi, proxy ayarlarının yapılması, IP dağıtımı ve ağ trafiğinin yönetilmesi gibi optimizasyon problemleri için DNA moleküllerinden yararlanılmaktadır. Ayrıca Mohammad ve arkadaşları asansör problemlerinin çözümü için DNA hesaplamanın kullanımı üzerine bir çalışma gerçekleştirmişlerdir [3].

Bütün bu çalışmaların, uygulamada gerçekleştirilmesi için mutlaka DNA bilgisayarların geliştirilmesi gerekmektedir. Çünkü teorik olarak ileri sürülen çalışmaların pratikte kullanımı çok zor ve pahalı olmaktadır. Bu nedenle, DNA hesaplama ile yapılan çalışmalar genellikle biyoloji laboratuarlarında gerçekleştirilmektedir.

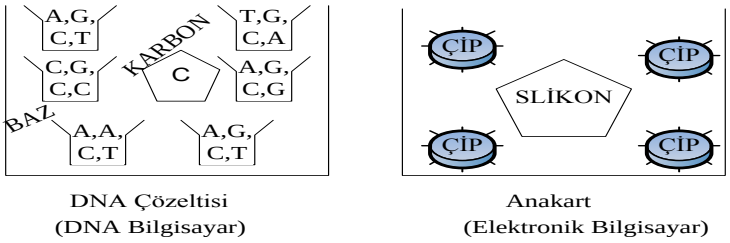
1.1. Moleküler Hesaplama ve Moleküler Bilgisayarlar

Bilgisayar bilimi, bilginin işlenmesi ve saklanmasıyla ilgilenmektedir. Bilginin işlenmesi konusunda, elektronik bilgisayarlar oldukça başarılı olmasına rağmen çok karmaşık ve zor problemlerde sıkıntı oluşturmaktadırlar. Bu sıkıntılar, moleküler hesaplama alanındaki gelişmeler sayesinde çözülecektir. Bu alanda bazı çalışmalar gerçekleştirilmiştir [1, 2, 32, 33]. Moleküler hesaplama, bilgisayar biliminde yeni ve uygulanabilir alanlardan biridir. Bu hesaplama, DNA veya proteinlerle yapılan bir hesaplama olup, paralel uygulamaların alt yapısını oluşturarak çok kısa zaman dilimlerinde milyonlarca işlemin yapılmasını sağlamaktadır. Bütün bu durumlara rağmen, moleküler hesaplama henüz yeni bir hesaplama alanıdır ve bu alanda çok büyük bir çözüm

potansiyeli bulunmaktadır. Biyoloji biliminin alt birimi olan genetik bilimi, bu alanda en çok kullanılan birimdir. Çünkü genetik bilgi, istenildiğinde saklanabilir, taşınabilir, birleşebilir, çoğalabilir ve başka verilere çevrilebilir. Genetik bilginin kullanılması ve işlenmesinde, nükleikasit ve protein yapıları kullanılmakta olup, bu yapılar DNA hesaplama için önemli bir role sahiptirler. Şuan ki araştırmalar, moleküler hesaplama yöntemiyle geleneksel hesaplama yöntemi arasında oluşturulacak bağ konusundadır. Yani, problemlerin moleküler hesaplamada kullanım şekli açıklanmaktadır.

Moleküler biyoloji, bilgisayar bilimi ve bilginin gelişmesine paralel olarak hızla gelişmektedir. Hücrelerin varlığı ve çalışma prensipleri bu alanda önemli bir yere sahiptir. Negatif enerji olarak isimlendirilen bu sistem genetik bilginin kod parçaları ile tutulduğunu göstermektedir. Bu alanda mikroskobik çalışmalar yürütülmüştür [32, 33].

Moleküler bilgisayarlar tasarlanırken, dijital bilgisayarlardaki silikon çipler yerine DNA moleküllerinin nanotüpler vasıtasıyla kullanılması üzerine çalışılmaktadır. Bu alandaki çalışmalardan biri Feynman [1] tarafından savunulan minyatür modelidir. Bu modelde amaç tamamen DNA moleküllerinden oluşan doğal bilgisayarlar yapmaktır. Şekil 1.6'da DNA bilgisayarlar ile elektronik bilgisayarların karşılaştırılması gösterilmektedir. Burada slikon yerine karbon molekülü, ana karttaki çipler yerine DNA moleküllerinin kullanımı amaçlanmaktadır.



Şekil 1.6. DNA bilgisayarlar ile elektronik bilgisayarların karşılaştırılması

Tablo 1.1. DNA ve elektronik bilgisayarların karşılaştırılması

DNA BİLGİSAYARLAR	ELEKTRONİK BİLGİSAYARLAR
Karbon	Silikon
Baz	Çip
Çözelti ortamı	Elektronik ortam
Moleküller	Transistörler ve kablolar
Veri saklama kapasitesi yüksek	Veri saklama kapasitesi düşük
Daha hızlı işlem yapma	Daha yavaş işlem yapma
Daha az elektrik kullanma	Daha çok elektrik kullanma

Moleküler bilgisayarları cazip hale getirecek diğer bir neden ise, onların güç tasarrufu sağlamasıdır. Çünkü, uygulamaların bazıları için çok büyük güç harcayan bilgisayarlara gerek yoktur. Bunun yerine, normal bilgisayarlardan çok daha az enerji harcayan bilgisayarlar da bu uygulamaları kolayca yapabilmektedir [1]. Tablo 1.1'de elektronik ve DNA bilgisayarların özellik olarak karşılaştırılması verilmektedir.

Moleküllerin bir başka kullanımı da farklı hesaplama yapılarını bir araya getirebilme yetenekleridir. Bugünkü bilgisayar yapısı, işlemci, hafıza, anakart, çip gibi elektronik elemanlardan oluşmaktadır. Bu yapının programlanması yeterince esnek olmamaktadır. Bununla beraber, biyolojik moleküller ile inşa edilen bir bilgisayar hem donanım hem de yazılım olarak yeterli esnekliğe sahip olmaktadır. Aynı zamanda, alan, zaman ve enerji açısından da oldukça verim sağlamaktadır.

Moleküller bir bilgiyi çeşitli yollarla elde edip işleyebilir. Bu yollara elektronların hareketleri örnek verilebilir. Elektronların hareketiyle yapılan bu işlemler moleküllerin hareketi ile de gerçekleştirebilir.

Kişisel ve dizüstü bilgisayarlar verileri saklarken bit kullanmaktadır. Bu bitler 0 veya 1 değeri alır. Bilgisayar programları da zaten 0 veya 1 dizilerinin gösterimidir. DNA molekülleri de bit gibidir. Bit 0 veya 1 değerlerinden biri ile ifade edilirken, DNA ise Adenin (A), Guanin (G), Sitozin (C), ve Timin (T) dörtlü bazını kullanmaktadır. Elektronik bilgisayarlardaki 0-1 dizilerinin yerine, DNA bilgisayarlarda A-T-G-C dizilerinden oluşan yapılar kullanılmaktadır.

Her hesaplama metodunda olduğu gibi, DNA hesaplamada da bazı hatalar ile karşılaşılmıştır. Bu durum, DNA hesaplamada zorlukların yaşanmasına neden olmuştur. Bundan dolayı, çoğu hesaplama metotlarında olduğu gibi DNA hesaplama metodu da % 100 güvenilir kabul edilmemiştir. Bu nokta da yapılması gereken en önemli iş, DNA dizilerinin çok dikkatli bir şekilde oluşturulmasıdır.

EMCC, DNA hesaplama alanındaki özel bir birlik olup 11 Avrupa ülkesinden oluşmaktadır [32]. Bu birliğin amacı, DNA hesaplama ve moleküler hesaplama alanında doküman hazırlamak ve gelişmeleri takip etmektir. Bu şekilde, araştırmacılara yeni ufuklar açıp onların bakış açısını değiştirmeyi hedeflemektedir. Bu kuruluş, henüz yeni olup çok hızlı bir şekilde gelişmesi beklenmektedir.

Avrupa Moleküler Hesaplama Birliği, Avrupa'daki gelişmeleri takip etmek için kurulmuştur. Bu kuruluş, 9 Avrupa Birliği ülkesindeki farklı araştırma kuruluşlarının bir araya gelmesiyle yeni bir fikir grubu olmuş ve daha çok bilim, teknoloji ve endüstri

alanlarında çalışma yapmıştır. Özel bir emek sarf edilirse, moleküler biyoloji, DNA hesaplama ve bilgisayar bilimleri hızla gelişecek ve istenilen düzeye ulaşacaktır. Bu bağlamda, EMCC Kurumu birçok ulusal ve uluslararası konferansı organize etmektedir.

Buraya kadar, DNA hesaplama ile ilgili teorik bilgi sunulmuş ve literatür anlatılmıştır. DNA hesaplamanın avantaj ve dezavantajları, zorlukları, uygulanabilirlik durumları ve hangi araştırmaların yapıldığı kısaca özetlenmiştir. Literatür çalışması Tablo 1.2'de özet şeklinde verilmektedir. DNA hesaplama üzerine yapılan çalışmalardan, DNA dizilerinin çok karmaşık ve zor problemlerin çözümünde ve çok büyük kapasitedeki verilerin saklanmasında, kolayca kullanılabileceği anlaşılmaktadır. DNA hesaplama ile ulaşılmak istenen asıl hedef, DNA moleküllerinden oluşan, DNA bilgisayarların yapılması ve geliştirilmesidir.

1.2. Literatür Özeti

Tablo 1.2'de DNA hesaplama algoritması kullanılarak yapılan çalışmaların özeti verilmektedir.

Tablo 1.2. DNA Hesaplama ile ilgili çalışmalar

Çalışmanın konusu	Çalışmanın Amacı	Kaynaklar
DNA Bilgisayarlar	DNA molekülü kullanarak DNA bilgisayar yapılabilmesi üzerine birçok çalışma yapılmıştır.	[1], [18], [28], [32], [33], [63]
Hamilton Path Problemi (Gezgin Satıcı Problemi)	Gezgin satıcı problemi birçok çalışmada kullanılmıştır.	[1], [2], [4], [6], [10],[13], [15]
Sırt Çantası Problemi	Belirli kapasitedeki bir kutu içerisine maksimum sayıda ve en fazla kazanç sağlayacak şekilde nesne yerleştirme işlemidir	[5], [8], [12], [71], [72]
3-SAT, Max-SAT ve Max-W-SAT problemleri	Mantıksal denklemlerin doğruluğunun sağlanması için kullanılır	[9], [11], [14],[20],[23]
Maksimum klikleme problemi	Bu tür problemlerde grafik üzerinde gezilecek maksimum düğümü bulma işlemi yapılır.	[16]

Tablo 1.2'nin devamı

Moleküler Biyoloji	Moleküler biyoloji ile DNA hesaplamanın ilişkisinin belirlenmesi	[59],[82]
Takagi-Sugeno Bulanık kurallar	Bulanık denetleyici parametrelerini DNA hesaplama ile ayarlamak	[17], [22], [58], [74]
Aritmetik Toplama	DNA molekülleri ile toplama işlemini gerçekleştirmek	[39]
Grafik Renklendirme Problemi	3 veya daha fazla renk kullanarak problemleri kodlamak ve çözümünü gerçekleştirmek	[24],[36],[64]
Minimum aralıklı ağaç problemi	Belirli bir bahçe veya alana minimum mesafe kullanarak en fazla ağaç dikimi yapmak	[7], [60]
PID parametrelerinin ayarlanması	PID parametreleri olan Kp, Ki ve Kd değerlerinin bulunması	[43], [45], [46], [47], [48], [52], [73]
Akıllı sistemler	Akıllı sistemleri oluşturan parametrelerin ayarlanması	[62], [63], [77], [78]
Sinyal işleme noktalarının belirlenmesi	Sinyal ayarlama için DNA hesaplamanın kullanılması	[38], [76], [79], [80]
Bilgi güvenliği ve şifreleme	DNA molekülleri ile güvenli veri merkezleri oluşturmak	[29], [30], [49], [70]
Olasılık parametre algoritması	DNA hesaplama ile olasılık hesaplamaları yapma	[81]
Ağ güvenliği	DNA hesaplama algoritması ile yasaklanmış permütasyon problemlerinin çözümü	[61], [65]
Matrikslerde çarpma	DNA hesaplama ile matriks çarpımını gerçekleştirmek	[75]
Asansör problemi	Asansör problemlerinin DNA hesaplama ile optimizasyonunu gerçekleştirmek	[3]
Veritabanı oluşturma	DNA molekülleri ile veri saklama işlemi yapılır.	[34], [35]

1.3. Tezin Amacı ve Kapsamı

Bu yüksek lisans tezinin amacı DNA hesaplama algoritmasının optimizasyon problemlerinin çözümünde, denetleyici parametrelerin ayarlanmasında ve sistem modellemede kullanılabileceğini göstermektir. DNA hesaplama son yıllarda kullanımı hızla artan bir optimizasyon algoritmasıdır. Yapılan uygulamalarda iki farklı DNA hesaplama kullanılmaktadır. Laboratuar ortamında yapılan DNA hesaplama biyokimyasal tepkimeler kullanılarak gerçekleştirilmektedir. Laboratuar uygulamasının maliyeti çok fazla ve uygulanabilirliği çok zordur. Bütün bu olumsuzlukları gidermek ve DNA moleküllerinin hesaplama yeteneğini kullanmak için çözelti ortamında uygulanan DNA hesaplamanın elektronik ortamda uygulanmasına yönelik çalışmalar yapılmaktadır. Bu tez çalışmasında çözelti ortamındaki DNA hesaplamanın elektronik ortamda uygulanabilecek özellikleri tespit edilerek sayısal DNA hesaplama algoritması geliştirilmektedir. Literatürde benzer örnekleri bulunan bu algoritma DNA hesaplama ile genetik algoritmanın benzer özelliklerinin kullanımı sonucu oluşturulmaktadır.

1.4. Tezin Yapısı

Bu tez çalışması 6 bölümden oluşmaktadır. Birinci bölüm giriş olup, bu bölümde DNA hesaplama hakkında genel bilgi verilmekte ve tarihçesi anlatılmaktadır. Ayrıca, DNA hesaplamaya ait literatür özeti, tezin amacı ve kapsamı geniş bir şekilde anlatılmaktadır.

İkinci bölümde, optimizasyon algoritmaları hakkında bilgi verilmekte özellikle, GA, QPSO ve YBS algoritmik adımları ile birlikte açıklanmaktadır.

Üçüncü bölümde, DNA hesaplama tüm yönleri ile ele alınmaktadır. DNA hesaplamanın, literatürdeki uygulama şekilleri, algoritmik adımları ve DNA hesaplamaya ait iki uygulamalı örnek bu bölümde verilmektedir.

Dördüncü bölümde, DNA hesaplama algoritması kullanılarak NP problemlerin optimizasyonu gerçekleştirilmektedir. Bu bölümde, DNA hesaplama iki optimizasyon problemine uygulanmakta ve optimum çözüm bulunmaktadır. Bu problemler; gezgin satıcı ve sırt çantası problemidir. Her iki problem de, küçük değerler kullanıldığında çözümü kolayca bulunan, fakat problemi oluşturan değişken sayısı arttıkça kompleks bir yapıya dönüşen, NP sınıfındandır. Bu kompleks problem yapısını çözmek için, elektronik bilgisayarlar kullanıldığında, matematiksel formülün oluşturulması çok zor olmaktadır.

Aynı zamanda, problemin boyutu büyüyeceği için, çözüm oluşuncaya kadar geçecek süre katlanarak artacaktır. Bu süre, problemin boyutuna göre haftalar, hatta aylar almaktadır. Bütün bu kompleks yapı içerisinde, çözümün bulunması zorlaşacaktır. Bunların bir sonucu olarak, problemin çözümünü elektronik bilgisayarla gerçekleştirdiğimizde, zaman, para ve enerji yönünden büyük dezavantaj yaşanmaktadır.

Gezgin satıcı, bir optimizasyon problemidir. Bu problemde, şehir sayıları ve bu şehirleri birbirine bağlayan yolların uzunlukları belirlenmekte ve bu değerler DNA molekülleri kullanılarak kodlanıp, DNA hesaplama kullanılarak, problemin optimizasyonu gerçekleştirilmektedir. Gezgin satıcı probleminde, her şehir en az ve sadece bir kez ziyaret edilmektedir. Bu bölümde, örnek bir gezgin satıcı problemi üzerinde, DNA hesaplama algoritması uygulanarak optimum çözüm bulunmaktadır.

Sırt çantası da aynı zamanda bir optimizasyon problemidir. Sırt çantası olarak adlandırılan bir nesne ve bu nesne içerisine yerleştirilecek elemanlar vardır. Her bir elemanın ağırlıkları ve kazanç değerleri farklıdır. Bu bölümde, belirli bir kapasitesi olan sırt çantası, örnek olarak alınmakta, sırt çantasına yerleştirilecek olan elemanların ağırlıkları ve kazanç değerleri belirlenmektedir. Problemin çözümü gerçekleştirilirken amaç, sırt çantası kapasitesini aşmadan en fazla kazanç değerini oluşturacak elemanların belirlenmesidir. Bu bölümde, örnek bir sırt çantası problemi üzerinde DNA hesaplama algoritması uygulanarak optimum çözüm bulunmaktadır.

Binaların ağ kablolarının döşenmesi işlemi, gezgin satıcı problemi ile aynı optimum çözüme sahiptir. Gezgin satıcı problemindeki şehirlerin yerine, bina içerisindeki odalar kullanılır. Burada amaç, en az uzunlukta kablo kullanarak, bütün odalara ağ kurmaktır. Bu problemin optimum çözümü ile gezgin satıcı probleminin optimum çözümü aynı algoritma ile bulunur.

Asansör problemi ise, sırt çantası problemine benzer bir çözüm yoluna sahiptir. Bu problemde, ağırlık sınırını geçmemek koşuluyla, maksimum nesnenin taşınması hedeflenmektedir. Bu yönüyle, tamamen sırt çantası problemine benzemektedir. Sırt çantası probleminin optimum çözüm yöntemi, aynen asansör problemi içinde kullanılmaktadır.

Tezin beşinci bölümünde, kontrol sistemlerine ait parametreler GA, QPSO ve DNA hesaplama algoritması ile ayarlanarak optimum çözüm gerçekleştirilmektedir. Bu bölüm, iki alt birimden oluşmaktadır.

İlk uygulama olarak, PI elemanlarının bir sistem üzerinde gösterimi, GA, QPSO ve DNA hesaplama ile optimizasyonu gerçekleştirilmektedir. PI elemanları, bir sistemin kararlı hale gelmesini sağlamaktadır. Sürekli kontrol edilmesi gereken ve sistemin çıkışına göre değer alan bu elemanlar, sistemde oluşacak hatanın minimum düzeyde tutulmasını sağlamaktadır. Sistemin kararlı halde kalması için, kullanılması gereken PI değerleri GA, QPSO ve DNA hesaplama ile bulunup, çevrimiçi ve çevrimdışı olarak sisteme uygulanmaktadır. Her bir algoritma için, çevrimiçi ve çevrimdışı değerler bulunmakta ve sonuçlar grafiksel olarak karşılaştırılmaktadır.

Bu bölümün ikinci uygulamasında GA, QPSO ve DNA hesaplama kullanılarak, bulanık denetleyicilerin ölçekleme faktörleri ayarlanmaktadır. Yukarıda belirtilen algoritmalar ile ölçekleme değerleri bulunup, sisteme gönderilmekte ve elde edilen sonuçlar grafik üzerinde incelenmektedir.

Tezin altıncı bölümünde, QPSO tabanlı uyarlamalı DNA hesaplama algoritmasıyla, PI parametrelerinin ayarlanması yapılmaktadır. Bu bölümde, DNA hesaplamaya ait popülasyon_boyutu, maksimum_işlem_sayısı, çaprazlama, enzim ve virüs oranı parametreleri, QPSO algoritması ile bulunarak DNA hesaplamaya gönderilmektedir. DNA hesaplama algoritması ise, sistemin hata ve çıkış değerlerini, QPSO algoritmasına göndermekte, burada uygulanan uygunluk fonksiyonu ile yeni parametre değerleri belirlenmektedir.

2. OPTİMİZASYON ALGORİTMALARI

Optimizasyon, bir problemin çözümü olabilecek değerler içerisinden en uygun olanı bulma işlemidir [40, 41]. Optimizasyonun amaçlarından biri matematiksel denkleme sahip bir problemin maksimum veya minimum değerini bulmak, analitik olmayan problemlerde ise en uygun analizi yapmaktır. Diğer bir amaç ise oluşabilecek hataları minimuma indirmek ve maksimum kazanç elde etmektir. Optimizasyonu, günlük hayatta farkında olarak veya olmayarak birçok kez kullanırız. Karşılaşılan bir problemde en uygun çözümü ararız. Örneğin, arabaların trafik lambalarında bekleme süresini en aza indirmek, bir imalat yapılırken en az malzeme ile maksimum kazanç elde etmek için gerekli hesaplamaları yapmak, optimizasyon problemlerinden sadece birkaçıdır. Optimizasyon problemleri, bütün koşullar düşünülerek çözülür. Sistemi kısıtlayan nedenler titizlikle irdelenir.

Optimizasyon problemlerinin uygulama alanlarından biri de NP problemlerdir. Bu tür problemlerde, değişken sayısı arttıkça problemin analitik çözümü zorlaşmaktadır. Örneğin gezgin satıcı probleminde şehir sayısı arttıkça, sırt çantası probleminde ise eleman sayısı arttıkça problemin çözümü zorlaşmakta ve çözüm için gerekli zaman uzamaktadır.

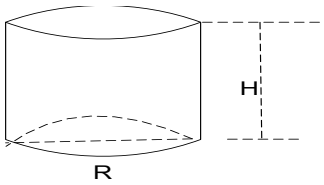
Bir optimizasyon probleminde, öncelikle 3 temel etkenin belirlenmesi gerekir. Bunlar; uygunluk fonksiyonu (fittness), değişkenler ve sınırlayıcılardır. Uygunluk fonksiyonu, çözümü istenen problemin optimizasyonunu bulmak için kullanılan matematiksel denklemdir. Optimizasyon işleminde uygunluk fonksiyonunu maksimum veya minimum yapan değerler bulunur. Denklem 2.1'de, örnek bir uygunluk optimizasyon problemi verilmektedir.

$$Max(y)= a_1x_1 + a_2x_2 + a_3x_3 + \ldots..+a_nx_n \quad (2.1)$$

Değişkenler, optimizasyon problemini oluşturan bağımsız parametrelerdir. Denklem 2.1'deki x_1, x_2, $x_{3\ldots..}x_n$ değişkenler olarak ifade edilir. Sınırlayıcılar ise, bir problemde değişkenler için belirlenen bir aralık olabileceği gibi, zamanla ilgili problemlerde süre sınırlaması şeklinde olabilir. Kısıtlayıcılar, problemin çözümü için bulunan optimum değerin çözüm olup olmadığını kontrol etmek içinde kullanılır. Optimizasyon problemlerinin çözümü için birçok algoritma geliştirilmiştir. Bunlardan bazıları; genetik

algoritma, DNA algoritması, parçacık sürü optimizasyon algoritması, yapay bağışık sistemler ve yapay sinir ağlarıdır.

Örnek bir optimizasyon problemi olarak seçilen Şekil 2.1'deki kutunun yüksekliği 2 cm ile 10 cm arasında, çapı 5 cm ile 12 cm arasında ve hacmi 250 cm^3 olacak şekilde imalatı yapılırsa toplam alanını minimum yapmak için uygunluk fonksiyonunu, değişkenleri ve sınırlayıcıları gösterelim.



Şekil 2.1. Örnek bir optimizasyon problemi

Değişken ve sınırlayıcılar ile alanın tanımlanması sonucu elde edilen amaç fonksiyonu denklem 2.2'de verilmektedir.

$$\min f(R,H) = \frac{\pi.R^2}{2} + \pi.R.H \tag{2.2}$$

Değişkenler Çap (R) ve Yükseklik (H) parametreleridir. Sınırlayıcılar denklem 2.3, 2.4 ve 2.5'te verilmektedir.

$$2 < H < 10 \tag{2.3}$$

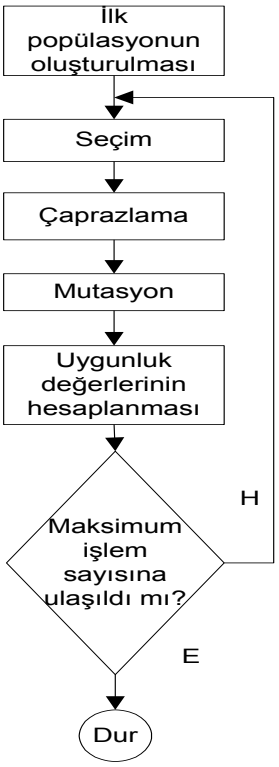
$$5 < R < 12 \tag{2.4}$$

$$V = 250cm^3 \tag{2.5}$$

2.1. Genetik Algoritma

Genetik Algoritma (GA), canlılardaki doğal seçim ve genetik yapıyı esas alarak geliştirilmiştir [40, 42]. Bilimsel araştırmalarda sıklıkla kullanılan bir optimizasyon

algoritmasıdır. Parametre değerleri iyi seçildiğinde, yerel çözüm noktalarına takılmadan global arama gerçekleştirebilirler. Paralel işlem yaparak daha hızlı sonuca ulaşırlar. Genetik bilgiyi kullanarak hesaplama yaparlar. Problemleri ikili sayı sistemi ile kodlarlar. Seçim, çaprazlama ve mutasyon özellikleri ile yeni popülasyon oluşturup matematiksel modellere gereksinim duymazlar. GA'da popülasyon büyüklüğü, seçim, çaprazlama ve mutasyon oranı problemin çözümü için bulunan sonuçları büyük oranda etkilemektedir. Bu parametrelerin iyi seçilmesi doğruluk oranını artırmaktadır. Ayrıca uygunluk fonksiyonunun iyi seçilmesi çok önemlidir. Şekil 2,2'de GA'nın adımları görülmektedir.



GA'nın sözle kodlanması:

Adım1: Rastlantısal olarak ikili sayı dizileri ile ilk popülasyonu oluştur.

Adım 2: Çözüm için belirli oranda eleman seç.

Adım 3: Yeni popülasyona çaprazlama uygula.

Adım 4: Popülasyona mutasyon uygula.

Adım 5: Popülasyon elemanlarının uygunluk değerlerini bul.

Adım 6: Maksimum işlem sayısına ulaşıldı ise dur değilse adım 2'ye geri dön.

Şekil 2.2. Genetik algoritma için akış diyagramı

Şekil 2.2'de akış diyagramı verilen GA'nın algoritmik adımları aşağıda açıklanmaktadır:

Adım 1: İlk popülasyon binary (ikili sayı sistemi) olarak ve rastlantısal bir şekilde oluşturulur. Bu adımda probleme ait tüm değişkenler gen olarak ifade edilen ikili sayı dizileri ile kodlanıp her bir dizi birleştirilerek kromozom olarak adlandırılan yapılar oluşturulur.

Adım 2: Problem için bir uygunluk fonksiyonu belirlenir. Popülasyona ait her bir eleman için uygunluk değerleri hesaplanarak kaydedilir.
Adım 3: Uygunluk değerleri en iyi olan elamanlardan uygulayıcı tarafından belirlenen bir oranda seçim yapılarak yeni popülasyon oluşturulur.
Adım 4: Seçilen elemanlar uygulayıcı tarafından belirlenen bir kurala göre çaprazlama işlemine tabi tutulur. Böylece ilk popülasyondan farklı yeni bir popülasyon oluşturulur. Yeni popülasyonun her bir elemanı için uygunluk değerleri hesaplanarak kaydedilir.
Adım 5: Yeni popülasyon uygulayıcı tarafından belirlenen bir oranda mutasyona tabi tutulur. Oluşan popülasyon elemanlarının uygunluk değerleri kaydedilir. Bu adıma kadar elde edilen uygunluk değeri en iyi popülasyonu belirler.
Adım 6: Yukarıda verilen 2, 3, 4 ve 5 adımları uygulayıcı tarafından belirlenen maksimum işlem sayısına ulaşılana kadar devam eder.
Adım 7: Maksimum işlem sayısına ulaşıldığında algoritma durdurulur. En iyi elemanlar seçilerek optimum çözüm elde edilir.

2.2. Kuantum Parçacık Sürü Optimizasyonu

Quantum parçacık sürü optimizasyonu (QPSO), doğadaki canlıların hareketlerinden esinlenerek geliştirilmiş bir algoritmadır. Bilimsel çalışma yapan kişiler, canlıların hareketlerini izleyerek onları anlamaya çalışıp, hareketleri üzerine detaylı çalışma yapmışlardır. Canlıların birbirleri ile olan iletişimi, birlikte hareket etme olgusu, dış etkilere ortak tepki verme durumu irdelenerek, bu olayların optimizasyon problemlerine uygulanabileceği sonucuna varılmıştır [43]. QPSO algoritması uygulanırken, her bir iterasyon için bireylerin kendilerinin sağladığı en iyi pozisyon pbest, bireylerin birlikte sağladığı en iyi pozisyon gbest değişkeninde tutulur. Parçacığın her bir iterasyon için hız değeri belirlenirken denklem 1.6 kullanılmaktadır.

$$v= (c1*p_{i.d} + c2*p_{g.d})/(c1+c2) \quad (2.6)$$

Denklem 2.6'da verilen c1 ve c2 değerleri [0 1] aralığında rastgele seçilen düzenli sayılardır. QPSO algoritmasının PSO algoritmasından farkı, sürünün elde ettiği gbest değeri yerine bu değerlerin ortalaması olan mbest değişkeni kullanılır. Mbest değeri denklem 2.7 kullanılarak hesaplanır.

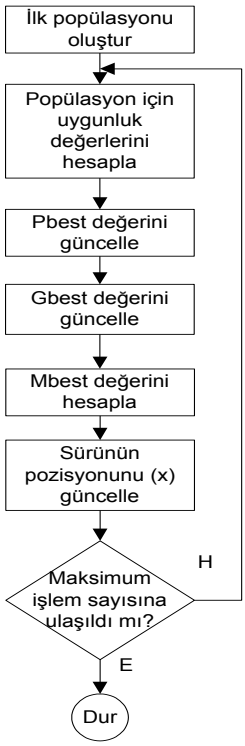
$$Mbest = 1/M * \sum(p_{g.d}(t)) \quad (2.7)$$

Denklem 2.7'deki M popülasyon büyüklüğünü ifade etmektedir. Popülasyonun yeni pozisyonu denklem 2.8 ve denklem 2.9 kullanılarak hesaplanır.

$$x(t+1) = p - \beta.|mbest - x(t)|.\ln(\frac{1}{u}) \qquad u<0.5 \quad (2.8)$$

$$x(t+1) = p + \beta.|mbest - x(t)|.\ln(\frac{1}{u}) \qquad u\geq 0.5 \quad (2.9)$$

Denklem 2.8 ve denklem 2.9'da kullanılan u ve β değişken olarak kullanılır. u değişkeni 0-1 aralığında bir değer alır. β değişkeni ise probleme göre kullanıcı tarafından belirlenir.



QPSO sözle kodlama

Adım 1: Rastlantısal olarak ilk popülasyonu oluştur.

Adım 2: Popülasyonun uygunluk değerlerini hesapla.

Adım 3: Her bir parçacığın Pbest değerini hesapla.

Adım 4: Pbest değerini kullanarak Gbest değerini hesapla.

Adım 5: Mbest değerini hesapla.

Adım 6: Bulunan yeni değerleri kullanarak sürünün yeni pozisyonu güncelle.

Adım 7: Maksimum işlem sayısına ulaşıldı ise dur değilse adım 2'ye geri dön.

Şekil 2.3. QPSO algoritması için akış diyagramı

Şekil 2.3'te akış diyagramı verilen QPSO algoritmasının algoritmik adımları aşağıda açıklanmaktadır:

Adım 1: İlk popülasyon rastlantısal olarak oluşturulur.

Adım 2: Problem için bir uygunluk fonksiyonu belirlenir. Popülasyona ait her bir eleman için uygunluk değerleri hesaplanarak kaydedilir.

Adım 3: Bulunan uygunluk değerleri parçacıkların yerel konum değerlerini verir. Her bir parçacığa ait bu yerel değerler karşılaştırılır ve uygunluk değeri en iyi olan parçacığın konumu, popülasyonun yeni konumu olarak alınır.

Adım 4: Popülasyonun global uygunluk fonksiyonu Mbest denklem 2.7 kullanılarak bulunur. Eğer bulunan değer önceki değerlerden iyi ise yeni değer Mbest olarak değiştirilir.

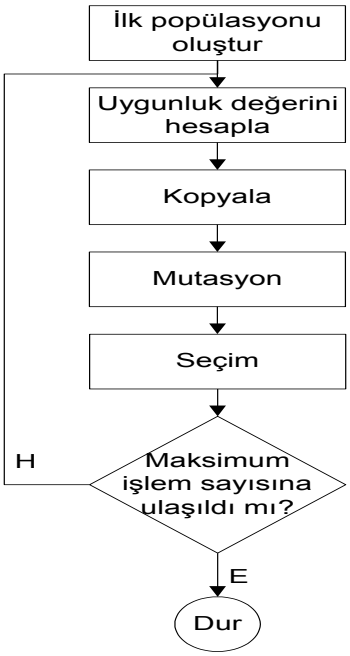
Adım 5: Popülasyonun en iyi hızı 2.6 denklemi kullanılarak hesaplanır.

Adım 6: Parçacıkların pozisyonu denklem 2.8 ve denklem 2.9 kullanılarak güncellenir.

Adım 7: Maksimum işlem sayısına ulaşılana kadar 2, 3, 4, 5 ve 6 adımları tekrarlanır. Maksimum işlem sayısına ulaşıldığında algoritma durdurulur. En iyi elemanlar seçilerek optimum çözüm elde edilir. Şekil 2.8'de QPSO algoritmasının adımları verilmektedir.

2.3. Yapay Bağışıklık Sistemler

Yapay Bağışıklık Sistemler (YBS), canlıların savunma mekanizması örnek alınarak geliştirilmiştir. Genetik algoritma ve DNA hesaplama algoritması gibi sezgisel yöntemler arasında sayılmaktadır [44]. Mühendislik problemlerinin çözümünde, çok iyi performans sağlamaktadır. YBS, algoritmik adımlar bakımından genetik algoritmaya benzemektedir. Mutasyon ve seçim her iki algoritma için ortak adımlardır. YBS, çaprazlama adımını kullanmayıp onun yerine kopyalama adımını kullanmaktadır. YBS ile yapılan uygulamalarda, bu algoritmanın lokal aramalarda daha etkili olduğu ancak global aramalarda yeterince performans sağlayamadığı görülmektedir. YBS, hafıza ve öğrenme gibi yetenekleri ile yapay sinir ağlarına benzemektedir. YBS ile yapılan uygulamalarda parametre seçimi çok önemlidir. Özellikle popülasyonun büyüklüğü, uygunluk değerinin belirlenmesi, maksimum işlem sayısı ve mutasyon oranının iyi belirlenmesi optimum çözüme ulaşmakta büyük öneme sahiptir. Genetik algoritma ve DNA Hesaplama algoritmasında olduğu gibi parametre seçimi uygulayıcının deneme yanılma ve tecrübelerine bağlı olarak farklılık göstermektedir. Ayrıca, problemin zorluk derecesi ve büyüklüğü de parametre seçiminde dikkate alınması gereken bir noktadır. Şekil 2.4'te YBS akış diyagramı verilmektedir.



YBS sözle kodlama

Adım 1: İkili sayı sistemi ile rastlantısal olarak ilk popülasyonu oluştur.

Adım 2: Popülasyonun uygunluk değerlerini hesapla.

Adım 3: Uygunluk değeri en iyi olanları kaydet.

Adım 4: Popülasyona belirlenen oranda mutasyon uygula.

Adım 5: Oluşan yeni popülasyondan uygunluk değeri en iyi olan elemanı belirle.

Adım 6: Maksimum işlem sayısına ulaşıldı ise dur değilse adım 2'ye geri dön.

Şekil 2.4. Yapay bağışık sistemler için akış diyagramı

Şekil 2.4'te akış diyagramı verilen YBS'nin algoritmik adımları aşağıda açıklanmaktadır:

Adım 1: İlk popülasyon rastlantısal bir şekilde oluşturulur.

Adım 2: Problem için bir uygunluk fonksiyonu belirlenir. Popülasyona ait her bir eleman için uygunluk değerleri hesaplanarak kaydedilir.

Adım 3: Uygunluk değerleri en iyi olan elamanlar kopyalanır.

Adım 4: Yeni popülasyon uygulayıcı tarafından belirlenen bir oranda mutasyona tabi tutulur. Oluşan popülasyon elemanlarının uygunluk değerleri kaydedilir. Bu adıma kadar elde edilen uygunluk değeri en iyi popülasyon belirlenir.

Adım 5: Yukarıda verilen 2, 3, 4 ve 5 adımları uygulayıcı tarafından belirlenen maksimum işlem sayısına ulaşılana kadar devam eder.

Adım 6: Maksimum işlem sayısına ulaşıldığında algoritma durdurulur. En iyi elemanlar seçilerek optimum çözüm elde edilir.

2.4. DNA Hesaplama

DNA hesaplama algoritması, son zamanlarda birçok optimizasyon probleminin çözümünde başarılı bir şekilde uygulanmaktadır. Özellikle, analitik yöntemlerle çözümü

güç olan problemlerin, DNA hesaplama algoritmaları kullanılarak çözüldüğü görülmektedir.

DNA hesaplama algoritması, Genetik algoritma ile benzerlik göstermektedir. Ancak, bu algoritma genetik algoritmalardan farklı olarak iki yeni mutasyon işlemine sahiptir [45, 46]. Enzim mutasyonu, herhangi bir DNA dizisinden bir veya daha fazla DNA parçasının silinmesi demektir. Enzim mutasyonu, sistem etkisiz hale geldiğinde yeni kontrol yapılarının seçimi için kullanılır. Virüs mutasyonu, herhangi bir DNA dizisine bir veya daha fazla yeni DNA parçası eklemektir. Virüs mutasyonu, sistem performansı olumsuz duruma geldiğinde kontrol yapılarının değiştirilmesi için kullanılır [47].

Şekil 2.5'de literatürde kullanılan DNA hesaplama akış diyagramındaki algoritmik adımlar aşağıda açıklanmaktadır [48]:

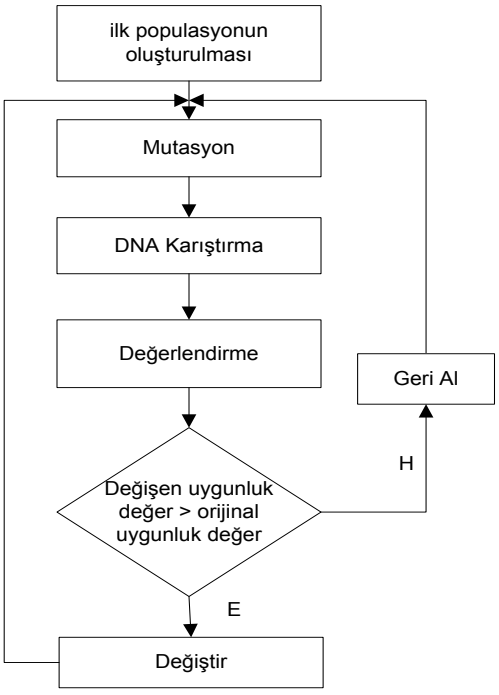
Adım 1: İlk popülasyon oluşturulur.

Adım 2: Popülasyona mutasyon uygulanır.

Adım 3: DNA dizileri karıştırılarak yeni popülasyon oluşturulur.

Adım 4: Popülasyonun uygunluk değerleri hesaplanıp değerlendirme yapılır.

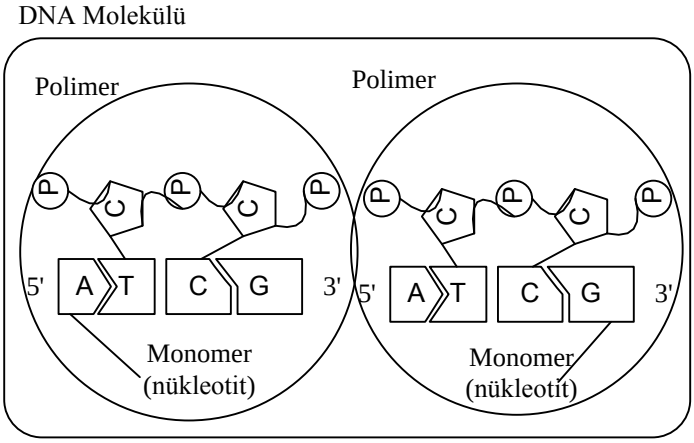
Adım 5: Yeni bulunan uygunluk değerleri orijinal değerlerden daha iyi ise değiştirilir ve adım 2'ye gidilir. Yeni değerler daha iyi değilse değiştirilmeden adım 2'ye gidilir.



Şekil 2.5. Literatürde kullanılan DNA hesaplama algoritması için akış diyagramı

3. DNA HESAPLAMA

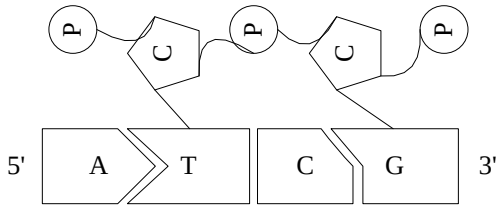
DNA hesaplama, son yıllarda kullanımı hızla artan yeni bir araştırma alanı olup, canlıların genetik bilgisini saklayan DNA moleküllerinin özelliklerinden yararlanılarak geliştirilmiştir [32, 49-55]. DNA, canlıların yaşamına dönük tüm bilgiyi kodlayan ve canlının yaşamı boyunca bozulmadan saklayan bir moleküldür. DNA molekülü, polimerlerden oluşmaktadır. Polimerler, monomer olarak isimlendirilen nükleotitlerin, yeterli sayıda birleşmesinden oluşmaktadır. Nükleotitler, deoksiriboz şeker, fosfat grubu ve bazlardan meydana gelmektedir [56, 57]. DNA molekülündeki karbonlar, molekülün yönünü belirlemektedir. DNA moleküllerinin pozitif yönü 5'–3' kalıbı ile, negatif yönü ise 3'–5' kalıbı ile gösterilmektedir. Molekülü oluşturan bazlar, Adenin, Guanin, Sitozin ve Timin olup sırayla A, G, C ve T harfleri ile ifade edilmektedir [58-64]. Şekil 3.1'de gösterildiği üzere, bu dört baz bir araya gelerek DNA moleküllerini oluştururlar. Nükleotitleri birbirinden ayıran tek etken, bazlar olduğu için, her bir nükleotit, baz isimleri ile ifade edilmektedir. Örneğin, Adenin nükleotiti yerine Adenin kullanılmaktadır.



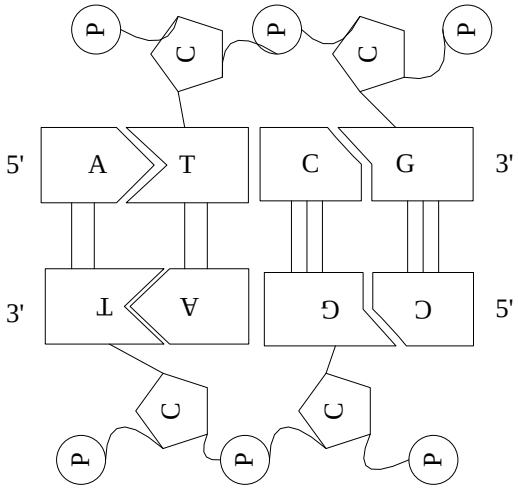
Şekil 3.1. DNA molekülünün kimyasal yapısı

DNA molekülü, yapısındaki hidrojen bağlarından dolayı kararlı bir yapıya sahiptir. Adenin ve Timin 2'li hidrojen bağı (A=T), Guanin ve Sitozin ise 3'lü hidrojen bağı (C≡G) kullanır. Guanin ve Sitozin üçlü hidrojen bağı nedeniyle Adenin ve Timin'e göre daha kararlıdır. DNA molekülleri tekli dizi ve çift sarmal dizi şeklinde bulunmaktadır. Tekli

DNA dizileri kullanılarak, DNA moleküllerinin sentezi ve çoğaltılması gerçekleştirilir. Şekil 3.2'de tek sarmal DNA ve Şekil 3.3'te çift sarmal DNA yapısı gösterilmektedir. DNA çift sarmal yapısı oluşurken, Watson-Crick tümleme kuralı kullanılır [52]. Bu kural, Watson ve Crick tarafından bulunmuş olup, Adenin ile Timin ve Guanin ile Sitozin birbirlerinin tümleyenidir [53, 54]. Çift sarmal DNA oluşturulurken, A ile T ve G ile C eşleşebilir. Bunun dışında bir eşleşme mümkün değildir. Bu yapıda paralellik olmayıp, DNA dizileri birbirlerinin zıt kutupları ile birleşir [55].



Şekil 3.2. Tekli DNA molekülünün genel yapısı



Şekil 3.3. Çift sarmal DNA molekülünün genel yapısı

Buraya kadar, DNA molekülünün genel yapısı ayrıntılı olarak anlatılmıştır. Şimdi ise DNA molekülünün hesaplamada kullanıldığında ne tür avantaj ve dezavantajları olduğu anlatılacaktır. DNA hesaplama, paralel işlem yapma yeteneği sayesinde çözümü yapılacak optimizasyon problemini alt gruplara ayırarak, işlemin daha hızlı ve daha kısa sürede gerçekleşmesini sağlar [32, 33]. İşlemlerde sağladığı hızın yanında, veri saklama kapasitesinin çok yüksek olması, DNA moleküllerinin elektronik veri saklama

aygıtlarından 10^{12} kat daha fazla veri saklayabilmesine olanak sağlar. Ayrıca, DNA molekülünün yıllarca bozulmadan kalması sayesinde, insanların genetik bilgisi uzun süre korunmaktadır. Uzun süreli korunması gereken verilerin saklanmasında, DNA moleküllerinden yararlanılmaktadır. Bütün bunların yanında, moleküllerin diziliminde oluşacak bir hatanın, hesaplama sonucunu doğrudan etkileme riski vardır. Bu şekilde bir hata yapıldığında sonuçlar çok farklı oluşmaktadır. DNA moleküllerinin hesaplama özelliklerinden azami fayda sağlanabilmesi için, uygulamaların çözelti ortamında gerçekleştirilmesi gerekmekte, bu ise daha fazla maliyet gerektirmekte ve uygulanabilirliği azaltmaktadır. Ayrıca bütün optimizasyon algoritmalarında olduğu gibi, DNA hesaplamada da bulunan sonucun doğruluk oranı %100 değildir. DNA hesaplamanın çözelti ortamında uygulaması zor ve pahalı olduğu için, son yıllarda yapılan çalışmalarda elektronik ortama uyarlanarak yeni sayısal DNA hesaplama algoritmaları geliştirilmiştir [45-48, 52]. Bu çalışmalarda amaç, DNA molekülüne ait özellikleri elektronik ortamda uygulamaktır. Yapılan birçok çalışmada, DNA moleküllerine ait özellikler kullanılarak bazı optimizasyon problemlerinin çözümü gerçekleştirilmiştir. Özellikle, literatürde çok kullanılan gezgin satıcı problemi, sırt çantası problemi ve denetleyici parametrelerinin ayarlanmasında DNA hesaplama başarılı bir şekilde uygulanmıştır [5, 7, 13, 15,52, 60, 61].

DNA hesaplama, literatürde optimizasyon algoritması olarak kabul edilmektedir. Optimizasyon algoritmaları, çözümü matematiksel yöntemlerle zor olan problemlerin optimum değerlerini bulmak için kullanılmaktadır. Polinomal olmayan zor (NP-hard) denklemler olarak kabul edilen bu tür problemlerin, parametre sayısı arttıkça zorluk derecesi ve karmaşıklık durumu artmakta, geleneksel algoritmalar ile çözümü zorlaşmaktadır [11, 50]. Bu nedenle, bu tür problemlerin çözümünün daha kolay gerçekleştirilmesi için sezgisel temellere dayanan algoritmalar geliştirilmiştir. Bu algoritmaların uygulamada en çok kullanılanları genetik algoritma, yapay bağışık sistemler, parçacık sürü optimizasyonu algoritması ve DNA hesaplama algoritmasıdır. Her algoritmanın kendine özgü algoritmik adımları vardır. Ayrıca, her algoritmanın yapılan uygulamalarda farklı problem türlerinde farklı performans sergilediği görülmektedir. Algoritmaların birbirlerine karşı avantaj ve dezavantajları olmakla birlikte algoritmik adımlarda benzerlik gösterenleri de vardır. DNA hesaplama algoritması, genetik algoritma ve yapay bağışık sistemler aynı temele dayandıkları için benzer özellikler göstermektedirler [13].

Tablo 3.1. DNA hesaplamanın diğer optimizasyon algoritmaları ile karşılaştırılması

Algoritma	Avantajları	Dezavantajları	Algoritmik adımlardaki farklılıklar	Çalışma hızı ve performansı
DNA Hesaplama	- Uygulanan mutasyon yöntemi ile daha geniş bir alanda çözüm kümesi oluşmaktadır. - Yerel optimum noktalara bağlı kalmaksızın genel arama yeteneğine sahiptir. - Karmaşık ve zor matematiksel denklem ve modellere gerek kalmaksızın çözüm elde edilmektedir. - Çözelti ortamında uygulandığında paralel işlem yapma yeteneği ile işlemleri daha hızlı gerçekleştirmektedir. - Çözelti ortamında uygulandığında büyük miktarda veri kodlama ve veri saklama kapasitesine sahiptir.	- Çözelti ortamında uygulaması zor ve pahalıdır. - Elektronik ortamda yapılan uygulamalarda DNA moleküllerinin hesaplama yeteneği tam olarak kullanılamamaktadır. - Diğer optimizasyon algoritmaları gibi elde edilen sonuçların %100 doğruluğu bilinmemektedir.	- Enzim ve Virüs mutasyonu -Sentez ve karıştırma	- Optimizasyon problemlerinde performansı çok iyidir. - Çözelti ortamında yapılan uygulamalarda paralel işlem yapma özelliği ile çalışma hızı çok yüksektir.
Genetik Algoritma	-Aritmetik yöntemlerle çözümü zor olan problemlerde kolaylıkla uygulanır. - Etkin arama yeteneği vardır. - Başlangıç çözümünden bağımsızdır. - Genetik bilgiyi kullanarak hesaplama yapar. - Karmaşık ve zor matematiksel model ve denklemlere gerek duymaz.	-Ayarlanması gereken parametre sayısı çok fazladır. -Mutasyon uygulaması ile problemin çözüm elemanının yok edilme riski vardır. -Belirli kısıtlamaları olan problemler için uygulanması zordur.	-Çaprazlama ve mutasyon	-Çalışma hızı iyidir. -Performansı problemin durumuna göre değişiklik göstermektedir.
QPSO	-Doğadaki canlıların hareketlerinden esinlenerek geliştirilmiştir. -Çok boyutlu problemlere kolaylıkla uygulanabilir. -Doğrusal olmayan karmaşık denklem kümesinin çözümünde başarılı bir şekilde uygulanır.	-Lokal aramalara takılıp kaldığı için global çözümlere ulaşması bazen zor olmaktadır. -Algoritması zordur. -Hesaplama yaparken bazı matematiksel denklemlere gerek duymaktadır.	-Parçacık hızının belirlenmesi -Sürünün yön vektörünün belirlenmesi	-Çalışma hızı DNA ve Genetik algoritmaya göre yavaştır. -Performansı probleme göre değişiklik göstermektedir.

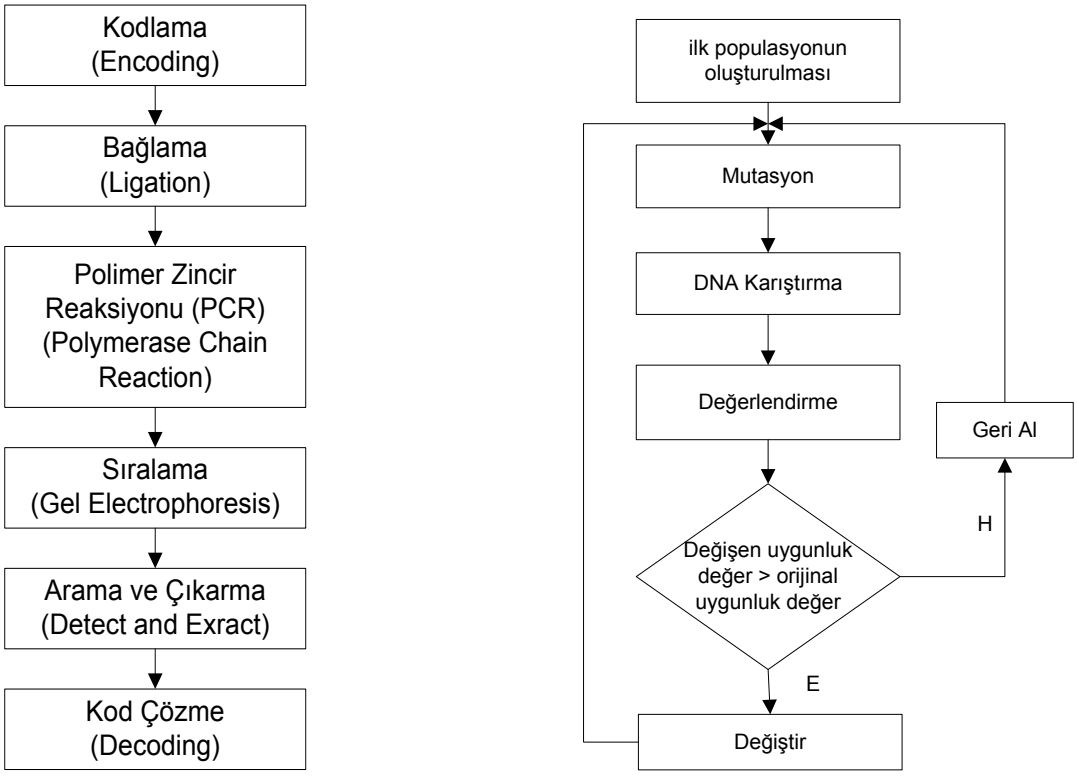
Tablo 3.1'in devamı

Yapay Bağışık Sistemler	-Canlılardaki savunma mekanizması örnek alınarak geliştirilmiştir. - Lokal aramalarda iyi performans sağlamaktadır. -Bazı optimizasyon problemlerinde diğer algoritmalara göre daha iyi performans sağlamaktadır.	-Global arama özelliği zayıftır. -Sistemin iyi performans sağlaması için ayarlanması gereken parametre sayısı fazladır.	-Genetik algoritmadan farklı olarak çaprazlama adımı kullanılmamaktadır. -Kopyalama adımı vardır.	-Çalışma hızı çok iyidir. -Performansı problemin durumuna göre değişiklik göstermektedir.

Tablo 3.1'de, literatürde sıklıkla kullanılan algoritmaların karşılaştırılması yapılmaktadır. Sezgisel temelli bu algoritmalar, genellikle zor matematiksel denklemlere gerek duymamakta, rastlantısal olarak oluşturulan bir çözüm alanı içerisinde arama yaparak sonuca ulaşmaktadırlar. Ayrıca, sezgisel temelli oldukları için, doğal hesaplama yöntemlerini kullanmaktadırlar. Bu algoritmaların en büyük dezavantajı, elde ettikleri sonuçların doğruluğundan %100 emin olunamaması ve her problemde aynı başarıyı gösterememeleridir.

Bu tez çalışmasında, DNA hesaplama, literatürdeki örneklerinden farklı olarak, elektronik ortama uyarlanarak kullanılmaktadır. Kullanılan DNA hesaplamanın, sayısal ortamdaki algoritmik adımları, Şekil 3.4'te ayrıntılı olarak gösterilmektedir. Yapılan uygulamalarda, DNA dizileri ile popülasyon oluşturulup sayısal değerlere çevrilerek, elektronik bilgisayarlarda kullanımı sağlanmaktadır.

Buraya kadar yapılan açıklamalarda, DNA hesaplamanın iki farklı uygulama şekli olduğu belirtilmektedir. Her iki DNA hesaplama algoritmasının, benzer ve farklı yönlerinin daha kolay anlaşılması için, Şekil 3.4'te literatürde kullanılan algoritmik adımlar ayrıntılı olarak verilmektedir.



a) Çözelti Algoritması b) Sayısal DNA hesaplama algoritma

Şekil 3.4. Farklı ortamlardaki DNA hesaplama algoritmalarının karşılaştırılması

3.1. Çözelti Ortamı Algoritması

Bugüne kadar, DNA hesaplama algoritması farklı şekillerde uygulanmıştır. Yapılan birçok çalışmada, bu algoritma çözelti ortamında gerçekleştirilmiştir [5, 8, 9, 61]. Çünkü DNA molekülünün hesaplama özelliği, çözelti ortamında gerçek şekilde uygulanabilmektedir. Bu uygulamalarda, biyokimyasal tepkimeler kullanılmaktadır. Bu tepkimelerde kullanılan algoritmik adımlar; hibritleşme (hybridization), onarma (ligation), polimer zincir reaksiyonu (polymerase cahin reaction – PCR), DNA dizilerinin sıralanması (gel electrophoresis), DNA dizilerinin ayrılması (melting-denaturing), DNA dizilerinin birleştirilmesi (annealing), DNA dizi parçasının çıkarılması (exract) ve özel dizi parçalarının aranarak bulunması (detect) olarak sıralanır. Çözelti ortamında hesaplama yapılırken, yukarıda sıralanan adımlardan hepsi veya bir kaçı kullanılmaktadır. Seçim yaparken, problemin karmaşık ve zorluk dereceleri ile algoritmik adımların uygulanabilirliği dikkate alınmaktadır. Çözelti ortamında uygulanan DNA hesaplama algoritması için kullanılan adımlar aşağıda verilmiştir.

Çözelti ortamı DNA hesaplama algoritması adımları şöyle açıklanabilir:

Adım 1: Kodlama (encoding). Problemin durumu dikkate alınarak kaç DNA bazı ile kodlanacağı ve eğer özel DNA dizi parçaları kullanılacaksa bunlar belirlenir. Belirlenen diziler, mikrodalga fırın büyüklüğünde sentez makineleri kullanılarak sentezlenir ve aynı özellikte DNA çözeltisi bulunan tüplere yerleştirilir. Dizi uzunluğunun ve kullanılacak dizilerin seçimi, problemin türü ve zorluk derecesi dikkate alınarak, uygulayıcı tarafından belirlenir. Literatür çalışmalarında, gezgin satıcı probleminin çözümü için yapılan çalışmalarda, bazı uygulayıcılar şehirlerin gösterimi için erime sıcaklıkları eşit olacak şekilde dizi uzunluğu belirlerken, yolların gösteriminde ise uzunluk ile erime sıcaklığının paralel olarak artması yöntemi kullanılmaktadır. Erime sıcaklığının (Tm) bulunmasında, uygulayıcılar farklı denklemler kullanmıştır. Bu denklemler, baz sayıları dikkate alınarak oluşturulmaktadır. Burada yapılan uygulamalarda, Tm değerinin hesaplanması için Denklem 3.1 kullanılmıştır [60].

$$Tm = 64.9 + [G_s + C_s - 16.4]/(G_s + C_s + T_s + A_s)*41 \quad (3.1)$$

Burada; A_s: Adenin sayısını, T_s: Timin sayısını, G_s: Guanin sayısını, C_s: Sitozin sayısını göstermektedir. Diğer değerler ise, uygulayıcının laboratuarda deneyimleri sonucu elde ettiği değerlerdir. Tablo 3.2'de, bir gezgin satıcı problemine ait şehirlerin, DNA dizileri ile kodlaması gösterilmektedir. Tablo 3.2 dikkatle incelendiğinde, şehirlerin her biri için erime sıcaklıkları aynı olacak şekilde, DNA dizileri seçilmektedir. Her şehir, 10 baz uzunluğunda DNA dizileri ile kodlanmaktadır. Bu sayı, uygulayıcı tarafından değiştirilebilir. Şehir sayısının az olduğu problemlerde, daha az sayıda baz kullanılabilir.

Tablo 3.2. DNA dizileri ile örnek kodlama

Şehir Adı	DNA dizileri ile gösterimi	Tm değeri
A	AGTGGGCCGG	20,2
B	CCGGACCGGA	20,2
C	CGGCCCTGTG	20,2
D	GGGATCACGC	20,2
E	TAGGGCCCGC	20,2
F	CCCGGGTTCG	20,2

Tablo 3.3'te, gezgin satıcı problemi için yol uzunluklarının DNA dizileri ile gösterimi verilmektedir. Tablo 3.3 dikkatlice incelendiğinde, yol uzunluğu ile erime sıcaklığı

değerleri paralel olacak şekilde seçilmiştir. Örneğin, uzunluğu 3 olan yolun erime sıcaklığı, uzunluğu 5 olan yoldan daha düşüktür.

Tablo 3.3. Yol uzunluklarının DNA dizileri ile gösterimi

Yol uzunluğu	DNA dizileri ile gösterimi	Tm değeri
3	TCTATAAAAT	1,76
5	AGTCTATACG	14,06
7	AGGATCCTGT	18,16
9	TACCCAGGCA	22,26

Adım 2: Bağlama (Ligase). Bu adımda, DNA dizileri ikişerli, üçerli veya daha fazla olarak birbirlerine bağlanarak, uzun DNA dizileri oluşturulur. Bu şekilde, problemin çözümü olabilecek tüm diziler elde edilir. Örnek dizilim, Tablo 3.4'te verilmektedir. Bu dizilim oluşturulurken, Tablo 3.2 ve Tablo 3.3'ten yararlanılmıştır.

Tablo 3.4. DNA bazları ile uzun dizilerin oluşması

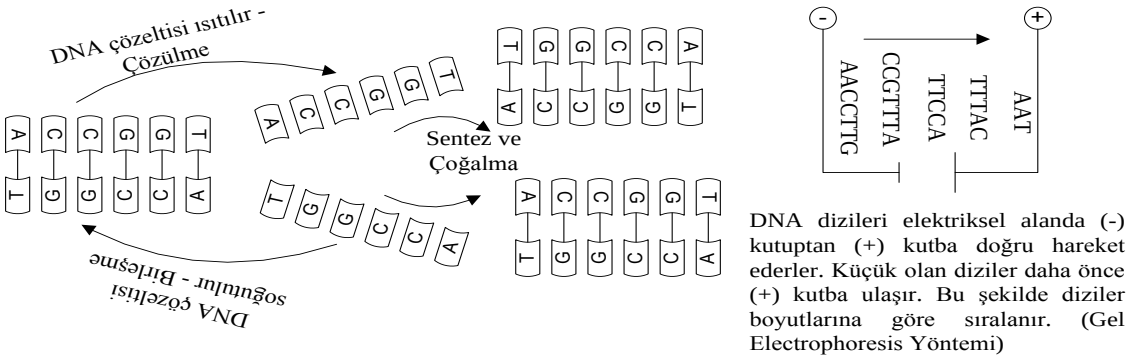
Şehirler	Yollar			Oluşan DNA dizisi
	Şehir 1	Mesafe	Şehir 2	
A - B	AGTGGGCCGG	TCTATAAAAT	CCGGACCGGA	AGTGGGCCGGTCTATA AAATCCGGACCGGA
A – B - C	AGTGGGCCGCT ATAAAATCCGG ACCGGA	AGTCTATACG	CGGCCCTGTG	AGTGGGCCGCTATAA AATCCGGACCGGA AGTCTATACG CGGCCCTGTG

Adım 3: Polymerase Chain Reaction (PCR). Polimer zincir reaksiyonu kullanılarak, çözüm için uygun olmayan elemanlar bulunup deney tüplerinden çıkarılır. Örneğin, bir şehre iki kez uğranılan bir yol varsa çözüm kümesinden çıkarılır.

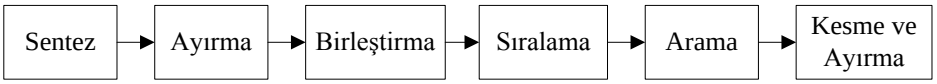
Adım 4: Gel Elecrophoresis (Sıralama). DNA dizileri, gel electrophoresis yöntemi kullanılarak, uzunluklarına göre sıralanır [62].

Adım 5: Arama ve Çıkarma. Sıralı durumdaki dizilerden, problemin türüne göre, en uzun veya en kısa olan dizi çözüm elemanı olarak belirlenip ayrılır. Dizi parçalarının kesilip çıkarılması için enzimler kullanılır. Yukarıda algoritmik adımları verilen çözelti ortamı DNA hesaplama algoritması, Şekil 3.5 ve Şekil 3.6'da özet olarak gösterilmektedir. Şekil 3.5'de, çift sarmal yapıdaki DNA dizilerinin ayrılarak, tekli DNA dizilerine dönüşümü ve tersi olayın gerçekleşmesi gösterilmektedir. Bu şekilde ayrıca, DNA dizilerinin elektriksel

alanda hareket ettirilerek, uzunluklarına göre sıralanması (gel electrophoresis) şematik olarak verilmektedir.



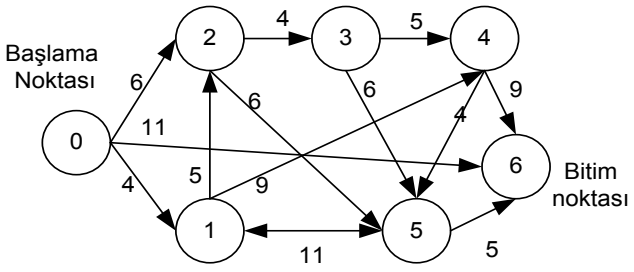
Şekil 3.5. DNA hesaplama algoritmasının çözelti ortamındaki adımları



Şekil 3.6. Çözelti ortamında DNA hesaplama algoritması

3.1.1. Çözelti Ortamı Uygulama Çalışması

Şekil 3.7'de verilen örnek problem için, çözelti ortamında DNA hesaplama algoritması adımları uygulanarak optimum sonuç elde edilmektedir.



Şekil 3.7. Örnek gezgin satıcı problemi

Adım 1: Şehirler ve şehirlerarasındaki yollar, DNA dizileri ile kodlanmakta ve her bir şehir ve mesafe 10 baz uzunluğunda DNA dizileri ile gösterilmektedir. Bu uzunluk şehir sayısı ve Tm değerlerinin durumuna göre değiştirilebilir.

Tablo 3.5. Şehirlerin DNA dizileri ile gösterimi

Şehir Adı	DNA dizileri ile gösterimi	Tm değeri
0	AGGCTGGGCC	30,46
1	ACCGGCCGGA	30,46
2	CCTGTCCGCG	30,46
3	TTACGCGCCC	30,46
4	AGCGGCAGCC	30,46
5	GCGCCAGGCT	30,46
6	CAGGCAGCCG	30,46

Şehirler kodlanırken Tm değerleri eşit olacak şekilde, yolların uzunlukları kodlanırken her bir yol uzunluğu Tm değeri ile orantılı olacak şekilde seçilir. Tablo 3.5'te şehirler için kullanılan Tm değerleri, Tablo 3.6'da ise mesafelerin gösterimi için kullanılan Tm değerleri verilmektedir.

Tablo 3.6. Yol uzunluklarının DNA dizileri ile gösterimi

Yol uzunluğu	DNA dizileri ile gösterimi	Tm değeri
4	TCTATAAAAT	1,76
5	AGTCTATACG	14,06
6	AGGATCCTGT	18,16
9	TACCCAGGCA	22,26
11	GGCACGACTG	26,36

Adım 2: Şehirlerarasında olabilecek bütün yollar DNA dizileri ile oluşturulmaktadır (Vi – Vj). Çözüm yolu üretilirken *s1w1s2* yöntemi kullanılmaktadır. 1. Şehir s1, 2. şehir s2 ve iki şehir arasındaki uzunluk w1 ile gösterilmektedir. Tablo 3.7'de şehirlerarasında olabilecek bütün yollar DNA dizileri kullanılarak kodlanmaktadır.

Tablo 3.7. Bütün yolların DNA dizileri ile oluşturulması

Yollar	Yolların oluşturulması	Yolların DNA dizileri ile gösterimi
Y1	0 – 1	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA
Y2	0 – 2	AGGCTGGGCC AGTCTATACG CCTGTCCGCG
Y3	1 – 2	ACCGGCCGGA TCTATAAAAT CCTGTCCGCG
Y4	1 – 6	ACCGGCCGGA GGCACGACTG CAGGCAGCCG
Y5	2 – 3	CCTGTCCGCG TCTATAAAAT TTACGCGCCC
Y6	3 – 4	TTACGCGCCC AGGATCCTGT AGCGGCAGCC
Y7	3 – 5	TTACGCGCCC TACCCAGGCA GCGCCAGGCT
Y8	4 – 5	AGCGGCAGCC TCTATAAAAT GCGCCAGGCT
Y9	5 – 6	GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y10	0 – 1 – 2	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG

Tablo 3.7'nin devamı

Y11	0 – 1 – 6	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA GGCACGACTG CAGGCAGCCG
Y12	1 – 2 – 3	ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC
Y13	2 – 3 – 4	CCTGTCCGCG TCTATAAAAT TTACGCGCCC AGGATCCTGT AGCGGCAGCC
Y16	3 – 5 – 6	TTACGCGCCC TACCCAGGCA GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y17	4 – 5 – 6	AGCGGCAGCC TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y18	0 – 1 – 2 – 3	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC
Y19	0 – 2 – 3 – 4	AGGCTGGGCC AGTCTATACG CCTGTCCGCG TCTATAAAAT TTACGCGCCC AGGATCCTGT AGCGGCAGCC
Y20	0 – 2 – 3 – 5	AGGCTGGGCC AGTCTATACG CCTGTCCGCG TCTATAAAAT TTACGCGCCC TACCCAGGCA GCGCCAGGCT
Y22	1 – 2 – 3 – 4	ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC AGGATCCTGT AGCGGCAGCC
Y23	1 – 2 – 3 – 5	ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TACCCAGGCA GCGCCAGGCT
Y24	2 – 3 – 4 – 5	CCTGTCCGCG TCTATAAAAT TTACGCGCCC AGGATCCTGT AGCGGCAGCC TCTATAAAAT GCGCCAGGCT
Y25	2 – 3 – 5 – 6	CCTGTCCGCG TCTATAAAAT TTACGCGCCC TACCCAGGCA GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y26	3 – 4 – 5 – 6	TTACGCGCCC AGGATCCTGT AGCGGCAGCC TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y27	3 – 5 – 5 – 6	TTACGCGCCC TACCCAGGCA GCGCCAGGCT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y28	4 – 5 – 5 – 6	AGCGGCAGCC TCTATAAAAT GCGCCAGGCT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y29	0 – 1 – 2 – 3 – 4	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TTACGCGCCC AGGATCCTGT
Y30	0 – 2 – 3 – 4 – 5	AGGCTGGGCC AGTCTATACG CCTGTCCGCG TCTATAAAAT TTACGCGCCC AGGATCCTGT AGCGGCAGCC TCTATAAAAT GCGCCAGGCT
Y31	1 – 2 – 3 – 4 – 5	ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC AGGATCCTGT TCTATAAAAT GCGCCAGGCT
Y32	1 – 2 – 2 – 3 – 5	ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TACCCAGGCA GCGCCAGGCT
Y33	2 – 3 – 4 – 5 – 6	AGGATCCTGT AGCGGCAGCC TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y34	2 – 3 – 3 – 5 – 6	ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TACCCAGGCA GCGCCAGGCT

Tablo 3.7'nin devamı

Y35	3 – 4 – 4 – 5 – 6	TTACGCGCCC AGGATCCTGT AGCGGCAGCC TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y36	0 – 1 – 2 – 3 – 4 – 5	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TTACGCGCCC AGGATCCTGT TCTATAAAAT GCGCCAGGCT
Y37	0 – 2 – 3 – 4 – 5 -6	AGGCTGGGCC AGTCTATACG CCTGTCCGCG TCTATAAAAT TTACGCGCCC AGGATCCTGT AGCGGCAGCC TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y38	1 – 2 – 3 – 4 – 4 – 5	ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC AGGATCCTGT TCTATAAAAT GCGCCAGGCT
Y41	0 – 1 – 2 – 3 – 4 – 5 – 6	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TTACGCGCCC AGGATCCTGT TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y42	0 – 2 – 3 – 4 – 5 – 5 – 6	AGGCTGGGCC AGTCTATACG CCTGTCCGCG TCTATAAAAT TTACGCGCCC AGGATCCTGT AGCGGCAGCC TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y43	0 – 1 – 2 – 3 – 4 – 5 – 6 – 1	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TTACGCGCCC AGGATCCTGT TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG TCTATAAAAT ACCGGCCGGA
Y44	0 – 1 – 2 – 3 – 4 – 5 – 6 – 2	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TTACGCGCCC AGGATCCTGT TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG AGTCTATACG CCTGTCCGCG
Y45	0 – 2 - 1 – 2 – 3 – 4 – 5 – 6	AGGCTGGGCC TCTATAAAAT TCTATAAAAT GCGCCAGGCT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TTACGCGCCC AGGATCCTGT TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG TCTATAAAAT ACCGGCCGGA
Y46	0 – 1 – 2 – 3 – 4 – 5 – 4 – 6	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TTACGCGCCC AGGATCCTGT AGGATCCTGT AGCGGCAGCC TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y47	0 – 1 – 2 – 3 – 4 – 5 –1 – 6	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TTACGCGCCC AGGATCCTGT TCTATAAAAT ACCGGCCGGA TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG
Y48	0 – 1 – 2 – 3 – 4 – 5 –5 – 6	AGGCTGGGCC TCTATAAAAT ACCGGCCGGA TCTATAAAAT CCTGTCCGCG TCTATAAAAT TTACGCGCCC TCTATAAAAT GCGCCAGGCT

Adım 3: Yukarıda rastgele üretilen çözüm içerisinde S_0 = AGGCTGGGCC (Şehir 0) ile başlayıp S_6 = CAGGCAGCCG (şehir 6) ile biten yollar olup olmadığı kontrol edilir. Bu duruma uymayan yollar çözüm havuzundan çıkarılır. Çünkü S_0 başlama noktası ve S_6 bitim noktası olarak seçilmiştir ve bulunacak çözümde olmak zorundadır. Bu durumda Y11, Y37,Y41, Y42, Y45, Y46, Y47 ve Y48 yolları dışında kalan yollar çözümden çıkarılır. Bu işlemin yapılması için PCR yöntemi kullanılır.

Adım 4: Yukarıdaki örnek 7 şehirden oluştuğu için içerisinde S0, S1, S2, S3, S4, S5 ve S6 şehirlerinin tamamı bulunan yollar alınır diğer yollar çözüm havuzundan çıkarılır. Çünkü gezgin satıcı olan kişi bütün şehirlere uğramak zorundadır. Bu durumda Y41, Y43, Y44, Y45, Y46, Y47 ve Y48 yolları dışında kalan yollar çözümden çıkarılır. Bu adımın uygulaması PCR yöntemi kullanılarak gerçekleştirilir.

Adım 5: Yukarıdaki yollardan her şehre sadece 1 kez uğrayan yollar çözüm için alınabilir. Eğer bir şehre iki kez veya daha fazla uğranıldı ise çözüm kümesinden çıkarılır. Bu durumda Y43, Y44, Y45, Y46, Y47 ve Y48 yolları çözümden çıkarılır ve sadece Y41 yolu çözüm için kalır.

Adım 6: Eğer en son kalan çözüm kümesinde birden fazla yol bulunsaydı kalan yolların Tm değerleri bulunur ve en küçük olan değer çözüm olarak alınırdı. Eğer Tm değerleri yerine DNA uzunlukları kullanılarak hesaplama yapılırsa adım 6 sonunda en kısa DNA dizisi çözüm olmak zorundadır. Çünkü örnek olarak seçilen problem minimumu bulmayı gerektirmektedir. En kısa uzunlukta DNA dizisi gel electrophoresis yöntemi ile belirlenir. Bu problemde tüm koşulları sağlayan Y41 yolu optimum çözüm olarak bulunmaktadır.

YOL-41: Y0-Y1-Y2-Y3-Y4-Y5-Y6 olup DNA dizilimi AGGCTGGGCC TCTATAAAAT TCTATAAAAT ACCGGCCGGA CCTGTCCGCG TCTATAAAAT TCTATAAAAT TTACGCGCCC AGCGGCAGCC TCTATAAAAT TCTATAAAAT GCGCCAGGCT TCTATAAAAT CAGGCAGCCG şeklindedir.

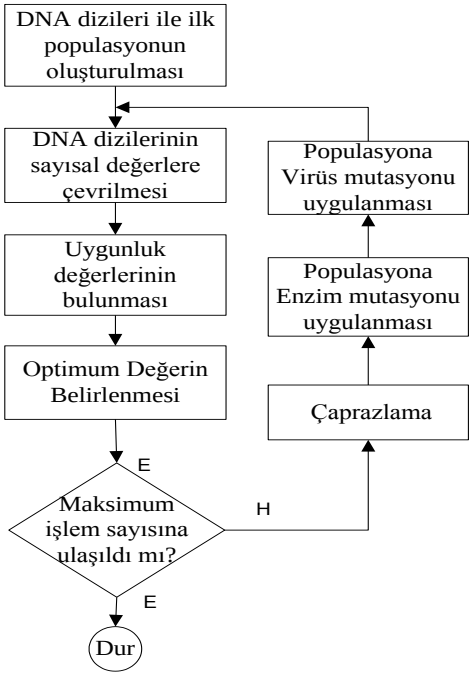
Çözüm olarak belirlenen Y41 yolu içerisindeki DNA bazlarının sayısı aşağıda verilmektedir. Bu değerler Denklem 3.1'de yerine konulduğunda Tm değeri bulunmaktadır. Baz sayıları aşağıda verilmektedir.

A: 44
G: 26
C: 36
T: 34

Tm= 64.9 + [(26+36 – 16.4)/(44+26+36+34)]*41
Tm= 78.25

3.2. Sayısal Ortam Algoritması

DNA hesaplamanın, çözelti ortamında uygulaması zor ve maliyetlidir. Bu nedenle son zamanlarda elektronik bilgisayarlarda DNA hesaplamanın kullanımına yönelik çalışmalar yapılmaktadır. Bu çalışmalarda DNA hesaplamanın genetik algoritma ile olan benzerlikleri kullanılmaktadır [45-48]. Bunlara ilave olarak DNA moleküllerine ait olan enzim ve virüs mutasyonu, DNA dizilerinin karıştırılması, DNA dizilerinin istenilen bölümlerinin tersten yazılarak yeni popülasyonların oluşturulması gibi algoritmik adımlarda kullanılmaktadır. DNA hesaplama genetik algoritmadan farklı olarak iki tür mutasyon uygulaması kullanmaktadır. Bunlar enzim ve virüs mutasyonudur. Enzim mutasyonu kullanılarak, popülasyondan belirli orandaki elemanların silinmesi sağlanır. Virüs mutasyonu kullanılarak ise popülasyona yeni elemanların eklenmesi sağlanır. Bu iki mutasyon kullanılarak, algoritmanın her bir adımı işletildiğinde popülasyonun yenilenmesi sağlanır. Böylece, DNA hesaplamanın yerel aramalardan kurtulup, genel arama yeteneği gelişmiş olur. Şekil 3.8'de, DNA hesaplama algoritmasının elektronik bilgisayarlarda kullanılan akış diyagramı verilmektedir. Elektronik bilgisayarlarda uygulanan DNA hesaplama algoritması için kullanılan adımlar aşağıda açıklanmaktadır.



DNA hesaplama algoritması sözle kodlama:

Adım 1: DNA dizileri ile rastlantısal olarak ilk popülasyonu oluştur.

Adım 2: DNA dizilerini sayısal verilere çevir.

Adım 3: Popülasyonun uygunluk değerini hesapla.

Adım 4: Optimum değeri belirle.

Adım 5: Maksimum işlem sayısına ulaşıldı ise dur değilse çaprazlama, enzim ve virüs mutasyonu uygula ve adım 2'ye geri dön.

Şekil 3.8. Sayısal ortam algoritması

Sayısal DNA hesaplama algoritması adımları:

Adım 1: Başlangıç popülasyonun DNA dizileri ile oluşturulması. Popülasyon DNA dizileri kullanılarak rastlantısal olarak belirlenir. Bu adımda popülasyon büyüklüğü ile popülasyonu oluşturan elemanların kaç adet DNA bazı ile gösterileceği uygulayıcı tarafından belirlenir. Popülasyon büyüklüğü problemin zorluğu ve çözüm hassasiyeti dikkate alınarak belirlenir. Popülasyon büyüklüğü arttığında daha geniş bir çözüm alanı oluşmakla beraber algoritmanın tamamlanması için daha uzun bir süreye ihtiyaç duyulur. DNA bazları ile kodlama yapılırken; A, G, C ve T olmak üzere 4 DNA bazı kullanılır ve sayısal dizilim oluşturulurken 4'lü sayı sistemi kullanılarak kodlama yapılır. Bu uygulamada A-0, G-1, C-2 ve T-3 olarak alınıp 4'lü sayı sistemi elde edilir. Örneğin AATCG dizisi 00321 sayısı ile 4'lü sisteme sonra onluk sisteme çevrilir. DNA dizileri ile kodlama ve sayısal değerlerin oluşturulması Tablo 3.8 ve Tablo 3.9'da ayrıntılı olarak gösterilmektedir.

Tablo 3.8. DNA hesaplamada kodlama yöntemi

	DNA Kod Sistemi	4'lü Kod sistemi	İkili kod sistemi
Kodlama	A, G, C, T	0, 1, 2, 3	00, 01, 10, 11

Tablo 3.9. DNA hesaplamada örnek kodlama

DNA ile kodlama	4'lü kodlama	Onluk Sistemdeki Değeri
GCC	122	1*16+2*4+2*1=26
GAG	101	1*16+0*4+1*1=17
GCA	120	1*16+2*4+0*1=24
GGT	113	1*16+1*4+3*1=23
GTA	130	1*16+3*4+0*1=28
GGA	110	1*16+1*4+0*1=20
GCA	120	1*16+2*4+0*1=24
GTA	130	1*16+3*4+0*1=28

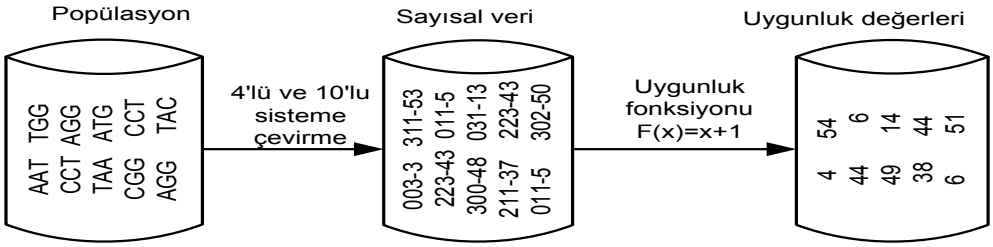
Tablo 3.8 ve Tablo 3.9 incelendiğinde DNA bazları tekli kullanıldığında en fazla 4 eleman, ikili kullanıldığında ise en fazla 16 eleman kodlanabilmektedir. Kodlama

sistemi 4'ün katı şeklinde artarak devam etmektedir. Bu durum Tablo 3.10'da ayrıntılı olarak gösterilmektedir.

Tablo 3.10. DNA baz sayısına göre kodlama

DNA baz sayısı	Kodlanabilecek değişken sayısı	Kodlanabilecek minimum değer	Kodlanabilecek maksimum değer	Örnek kodlama
1	4	0	3	A, G, C, T
2	16	0	15	AA, TA, CG
3	64	0	63	AAT, GGC
4	256	0	255	AATG, TCCA

Adım 2: Popülasyonu oluşturan elemanların sayısal değerlere çevrilerek uygunluk değerlerinin hesaplanması ve en iyi değerin belirlenmesi. DNA dizilerinin sayısal değerlere çevrilmesi adım 1'de ayrıntılı olarak gösterilmektedir. Uygunluk fonksiyonu problemin çözümünü gerçekleştirmek için kullanılan bir denklemdir. Her problem için farklılık gösterir. Problemin türüne göre uygulayıcı tarafından belirlenir. Popülasyonu oluşturan her eleman uygunluk fonksiyonunda yerine konularak uygunluk değerleri bulunur. Şekil 3.9'da örnek hesaplama verilmektedir.



Şekil 3.9. DNA hesaplamada dizilerin sayısal verilere dönüştürülmesi

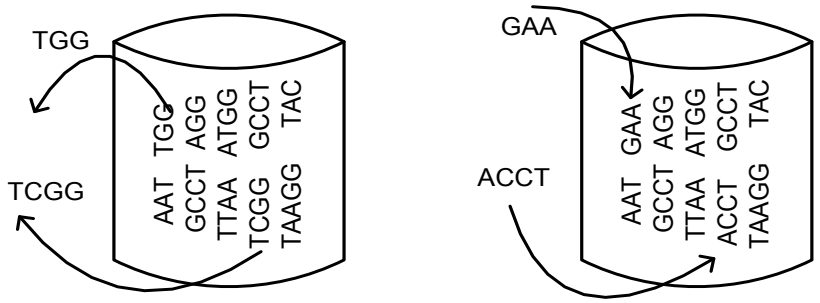
Adım 3: Çaprazlama ile yeni popülasyonun oluşturulması. Çaprazlama işlemi mevcut popülasyon elemanlarını ikişerli veya farklı şekillerde kullanarak yeni popülasyonun belirlenmesi işlemidir. Çaprazlama belirli kurallar ve uygulanacak oran belirlenerek gerçekleştirilir. Çaprazlama gerçekleştirilirken her bir elemanın hangi bölümlerinin çaprazlama için kullanılacağı uygulayıcı tarafından belirlenir. Tablo 3.11'de örnek bir oluşum gösterilmektedir.

Tablo 3.11. DNA hesaplamada örnek çaprazlama

1. Dizi	2. Dizi	Çaprazlama sonucu oluşan diziler
*AA*TG	*TG*CA	*AATG*, TGCA
*GA*CT	*TC*AG	*GATC*, CTAG
*AATG*CG	*GC*GCTA	*AATGGC*, CGGCTA

Örneğin, popülasyon elemanları AATG ve CCGT olsun. Bazı uygulayıcılar *AA*TG-*CC*GT →*AACC*, TGGT şeklinde çaprazlama yaparken, bir kısım uygulayıcılar ise *A*ATG-*CCG*T →*ACCG*, ATGT şeklinde çaprazlama yapabilir.

Adım 4: Popülasyona enzim ve virüs mutasyonu uygulanması. Uygulayıcı tarafından belirli bir enzim ve virüs oranı belirlenir. Bu oran problem türüne göre değişiklik gösterebilir. Belirlenen oranda elemanlar rastgele seçilerek popülasyondan çıkarılarak enzim mutasyonu gerçekleştirilir. Yine belirlenen oranda yeni eleman popülasyona eklenerek yeni bir çözüm alanı oluşturulur. Şekil 3.10'da görüleceği üzere önce enzim mutasyonu ile belirli oranda eleman popülâsyondan silinir. Silinen elemanların yerine aynı oranda yeni elemanlar eklenerek virüs mutasyonu gerçekleştirilir..



a) Enzim mutasyonu b) Virüs mutasyonu

Şekil 3.10. Enzim ve virüs mutasyonunun gerçekleşmesi

Adım 5: Yeni popülasyonun uygunluk değerlerinin hesaplanması ve en iyi değerin belirlenmesi. Burada, adım 2'ye benzer bir yöntem uygulanarak, uygunluk değerleri elde edilir. Bu adımda, belirlenen değerler kaydedilerek her bir iterasyonda karşılaştırma işlemi için kullanılır.

Adım 6: Maksimum işlem sayısı kadar yukarıdaki 2, 3, 4 ve 5 adımlarının tekrarlanması. Maksimum işlem sayısı, kullanıcı tarafından belirlenen ve döngünün kaç kez çalıştırılacağını belirleyen bir değerdir.

Adım 7: Maksimum işlem sayısına ulaşıldığında, sistemi optimum yapan parametrelerin belirlenmesi.

3.2.1. Sayısal Ortam Uygulama Çalışması

Sayısal DNA hesaplama algoritmasının uygulanması için, optimum değeri matematiksel yollarla kolayca bulunabilecek bir fonksiyon olan Denklem 3.2 seçilerek, elde edilen sonuçların gerçek değerler ile kontrol edilmesi sağlanmaktadır.

$$F(x)=26*x-x^2+192 \quad (3.2)$$

Bu bölümde yapılan uygulamada, Denklem 3.2'de verilen fonksiyonunun, $0 \leq x \leq 63$ aralığındaki optimum çözümü bulunacaktır. Problemde tek değişken x olup, 0 ile 63 arasında değer almaktadır. 0 ile 63 Aralığındaki değerleri 3 DNA bazı ile gösterebiliriz. Dört DNA bazı olup bu çalışmada A-0, G-1, C-2 ve T-3 olarak seçilmekte ve 4'lü sayı sistemine çevrilmektedir. AAA dizisi 0 değeri için, TTT dizisi ise 63 değeri için kullanılmaktadır. DNA bazlarının hangi sayısal değerler ile gösterileceği ve dizi uzunluğu, uygulayıcı tarafından belirlenir.

1.İterasyon:

Adım 1: Random olarak ilk popülasyonun oluşturulması için, DNA dizileri seçilmektedir. Tablo 3.12'de, seçilen DNA dizileri ve sayısal değerleri gösterilmektedir.

Tablo 3.12. Sayısal DNA hesaplama için ilk popülasyonun oluşturulması

DNA dizisi	4'lü değeri	Onluk değeri	Uygunluk değeri
AAT	003	3	**261 (Fonksiyonun aldığı mak. değer)**
TTC	332	62	-2040
TAA	300	48	-864
GTT	133	31	37
GTC	132	30	72
CCT	223	43	-539

Adım 2: Popülasyonun elemanlarından rastgele seçim yapılarak, çaprazlama işlemi gerçekleştirilir. Bu adımda, 1-2, 3-4 ve 5-6 elemanları birbirleri ile çaprazlanmaktadır. Tablo 3.13'te, çaprazlanan elemanlar ve çaprazlama sonucu oluşan popülasyon verilmektedir.

Tablo 3.13. Birinci iterasyon için popülasyona çaprazlama uygulanması

Çaprazlanan elemanlar	Çaprazlama sonucu oluşan yeni elemanlar	4'lü değeri	Onluk değeri	Uygunluk değeri
*A*AT	A*TC*	032	14	**360(Fonksiyonun aldığı mak. değer)**
*T*TC	T*AT*	303	51	-1083
*T*AA	T*TT*	333	63	-2139
*G*TT	G*AA*	100	16	352
*G*TC	G*CT*	123	27	165
*C*CT	C*TC*	232	46	-728

Adım 3: Bu adımda, popülasyona enzim ve virüs mutasyonu uygulanmaktadır. Mutasyon uygulanacak elemanlar rastlantısal olarak ve daha önce belirlenen oranda seçilmektedir. Enzim ve virüs mutasyonu ile 3. eleman TTT silinerek GAG eklenmekte, 6. eleman CTC silinerek CCT eklenmekte ve yapılan uygulama Tablo 3.14'te gösterilmektedir.

Tablo 3.14. Birinci iterasyon için popülasyona enzim ve virüs mutasyonu uygulanması

Enzim mutasyonu (rastgele değer sil)	Virüs mutasyonu (yeni değer ekle)	4'lü değeri	Onluk değeri	Uygunluk değeri
ATC	ATC	032	14	**360 (Fonksiyonun aldığı mak. değer)**
TAT	TAT	303	51	-1083
TTT	*GAG*	*101*	*17*	*345*
GAA	GAA	100	16	352
GCT	GCT	123	27	165
CTC	*CCT*	*232*	*46*	*-728*

2. İterasyon:

Adım 1: Popülasyonun elemanlarından rastgele seçim yapılarak, çaprazlama işlemi gerçekleştirilir. Bu adımda, 1-3, 2-5 ve 4-6 elemanları çaprazlanmaktadır. Tablo 3.15'te, çaprazlanan elemanlar ve çaprazlama sonucu oluşan popülasyon verilmektedir.

Tablo 3.15. İkinci iterasyon için popülâsyona çaprazlama uygulanması

Çaprazlanan elemanlar	Çaprazlama sonucu oluşan yeni elemanlar	4'lü değeri	Onluk değeri	Uygunluk değeri
*A*TC	AAG	001	1	217
*T*AT	TCT	323	59	-1755
*G*AG	GTC	132	30	72
*G*AA	GCT	123	27	165
*G*CT	GAT	103	19	**325 (Fonksiyonun aldığı mak. değer)**
*C*CT	CAA	200	32	0

Adım 2: Popülasyona, enzim ve virüs mutasyonu uygulanarak, yeni elemanlar oluşturulmaktadır. Oluşan yeni elemanların, uygunluk değerleri tekrar hesaplanmaktadır. Enzim ve virüs mutasyonu ile 2. eleman TCT silinerek TAA eklenmekte, 3. eleman GTC silinerek GGC eklenmekte ve yapılan uygulama Tablo 3.16'da gösterilmektedir.

Tablo 3.16. İkinci iterasyon için popülasyona enzim ve virüs mutasyonu uygulanması

Enzim mutasyonu (rastgele değer sil)	Virüs mutasyonu (yeni değer ekle)	4'lü değeri	Onluk değeri	Uygunluk değeri
AAG	AAG	001	1	217
TCT	*TAA*	300	48	-864
GTC	*GGC*	122	26	192
GCT	GCT	123	27	165
GAT	GAT	103	19	**325 (Fonksiyonun aldığı mak. değer)**
CAA	CAA	200	32	0

3. İterasyon:

Adım 1: Popülasyonun elemanlarından rastgele seçim yapılarak, çaprazlama işlemi gerçekleştirilir. Bu adımda, 1-4, 2-6 ve 3-5 elemanları çaprazlanmaktadır. Tablo 3.17'de çaprazlanan elemanlar ve çaprazlama sonucu oluşan popülasyon verilmektedir.

Tablo 3.17. Üçüncü iterasyon için popülasyona çaprazlama uygulanması

Çaprazlanan elemanlar	Çaprazlama sonucu oluşan yeni elemanlar	4'lü değeri	Onluk değeri	Uygunluk değeri
AAG	ACT	013	7	325
TAA	TAA	300	48	-864
GGC	GAT	103	19	325
GCT	GAG	101	17	**345 (Fonksiyonun aldığı mak. değer)**
GAT	GGC	112	22	280
CAA	CAA	200	32	0

Adım 2: Popülasyona, enzim ve virüs mutasyonu uygulanarak, yeni elemanlar oluşturulmaktadır. Enzim ve virüs mutasyonu ile 1. eleman ACT silinerek ATG eklenmekte, 6. eleman CAA silinerek GAA eklenmekte ve yapılan uygulama Tablo 3.18'de gösterilmektedir.

Tablo 3.18. Üçüncü iterasyon için popülasyona enzim ve virüs mutasyonu uygulanması

Enzim mutasyonu (rastgele değer sil)	Virüs mutasyonu (yeni değer ekle)	4'lü değeri	Onluk değeri	Uygunluk değeri
ACT	*ATG*	*031*	*13*	**361 (Fonksiyonun aldığı mak. değer)**
TAA	TAA	300	48	-864
GAT	GAT	103	19	325
GAG	GAG	101	17	345
GGC	GGC	112	22	280
CAA	*GAA*	*100*	*16*	*352*

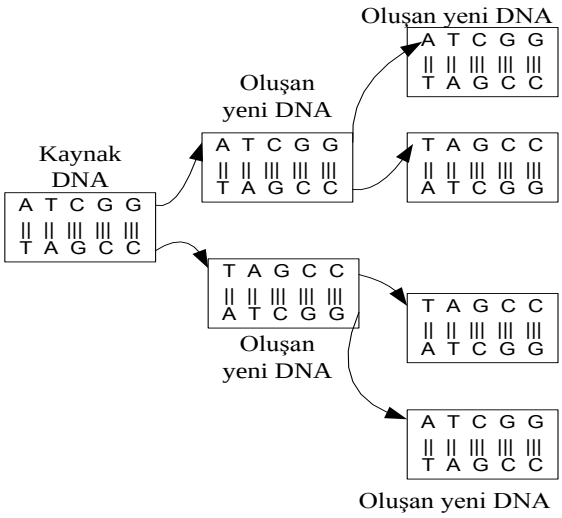
Yukarıda, 3 iterasyonu verilen problemde, fonksiyonun alabileceği maksimum değer x=13 noktasında olmaktadır. Bu değerden daha büyük veya daha küçük x değerleri için,

fonksiyon daha küçük değerler almaktadır. Denklem 3.2 için uygulanan sayısal DNA hesaplama algoritması sonucu, fonksiyonun optimum değeri x= 13 noktasında ve 361 olarak bulunmaktadır.

3.3. DNA Hesaplama için Kullanılan Metotlar

3.3.1. DNA Dizilerinin Sentezlenmesi

DNA dizilerinin çoğaltılarak milyonlarca kopyasının oluşturulmasına, sentez denilir. DNA dizilerinin laboratuar ortamında sentezlenmesi için, mikrodalga fırın büyüklüğünde sentez makineleri kullanılmaktadır. Problemlerde kullanılmak istenen DNA dizilerini, istenilen kalıplarda üretmek mümkündür. Bunun için, kullanıcının örnek bir DNA dizisini, sentez makinesine koyması yeterli olmaktadır. Çünkü DNA dizileri, bir kalıptan Watson – Crick tümleme kuralını kullanarak çoğalabilmektedir [32, 33]. DNA dizileri çoğaltılmadan önce, çift sarmal yapı ısıtılarak, tek sarmal DNA dizilerine dönüştürülür. Şekil 3.11'de, bu olay şematik olarak gösterilmektedir. Her bir tekli dizi, DNA dizilerinin çoğaltılası için kalıp görevi görür. Oluşan bu diziler, yeni bir kalıp olarak kullanılır ve çok kısa sürede DNA dizilerinin sayısı, milyonları hatta milyarları bulur. Sentez işlemi, problemlerin çözüm kümesini oluşturmak için kullanılmaktadır. Kullanıcı tarafından verilen T_1, T_2, T_3,…..T_n tüplerinde sentezi gerçekleştirilen DNA dizileri, tek bir tüpte toplanarak birleştirilir. Şekil 3.11'de, örnek bir DNA sentezi verilmektedir.

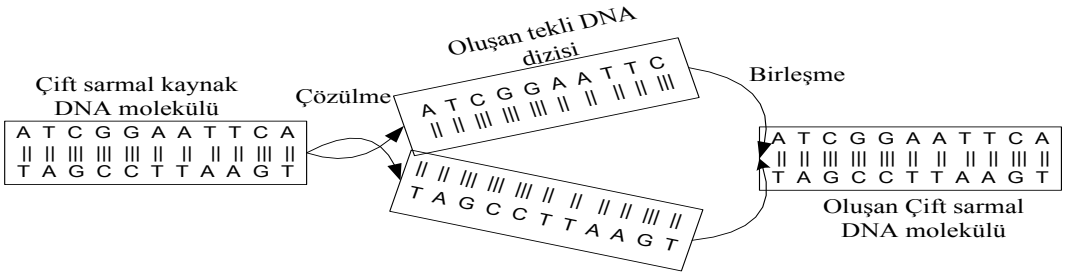


Şekil 3.11. DNA sentezinin gerçekleşmesi

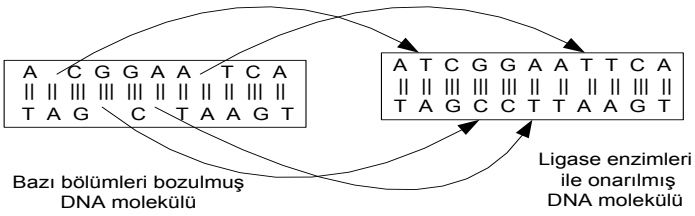
3.3.2. DNA Dizilerinde Ayrılma, Birleşme ve Onarma İşlemi

DNA çözeltilerini belirli bir sıcaklık değerine kadar ısıtırsak, çift sarmal halde bulunan DNA'lar çözülerek, tekli DNA dizilerine dönüşür. Bu şekilde çözülme işlemi gerçekleşmiş olur. Çözeltinin ısıtılması sırasında, bazlar arasında bulunan hidrojen bağları koparak, çift sarmal yapıyı oluşturan DNA dizilerinin, birbirlerinden ayrılması sağlanır. G-C ikilisi arasında 3'lü hidrojen bağı olduğu için bunların ayrılması, A-T ikilisine göre daha zor olmakta ve bu nedenle DNA dizilerinde C/G oranı fazla olan çözeltiler, daha kararlı olmaktadır [63-65]. DNA ile hesaplama işlemlerinde diziler oluşturulurken, bu özellik kullanılmakta ve C/G oranı değişkenler arasında eşit tutulmaya çalışılmaktadır.

DNA dizilerini oluşturan çözeltiler soğutularak, tekli DNA dizilerinin birbirleriyle tekrar bağlanıp, çift sarmal DNA yapısını oluşturması olayına, yeniden birleşme denilmektedir. Birleşme olayı, çözülme olayının tam tersi bir işlemdir [67-70]. Şekil 3.12'de, DNA dizilerinin çözülmesi ve tekrar birleşmesi gösterilmiştir. DNA dizilerinin birleşmeleri sırasında, herhangi bir nedenden dolayı birleşemeyen bazlar, *Ligase* denilen DNA enzimleri sayesinde onarılarak, işlemin tamamlanması sağlanır. Bu olaya, onarım denilmektedir. Şekil 3.13'te, DNA dizilerinin onarım olayı gösterilmektedir. Bu enzimler, çoğunlukla yeniden birleşme ve melezleşme olaylarında kullanılmaktadır.



Şekil 3.12. DNA dizilerinin çözülmesi ve birleşmesi



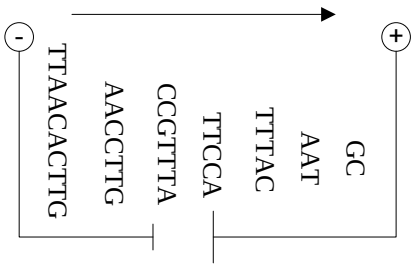
Şekil 3.13. DNA dizilerinin onarılması

3.3.3. DNA Dizilerinin Melezleşme ile Ayrılması

Melezleşme ile ayırma işlemi, DNA dizilerinin bulunduğu tüplerden, istenilen özellikteki özel DNA dizi parçasını ayırma işlemidir. Bu işlemin gerçekleştirilmesi sırasında, öncelikle ayırmak istediğimiz DNA parçasının tümleyeni olan dizi parçasının kopyası çoğaltılarak, DNA tüplerine konur. Bu kopyalar, tüp içerisinde bulunan ve bizim ayırmak istediğimiz parçanın tüm kopyasını bulup ayırırlar. Melezleşme olayı gerçekleşirken, DNA çözeltisi belirli bir sıcaklık değerine kadar soğutulur [66].

3.3.4. DNA Dizilerinin Sıralanması

Bu metot, DNA dizilerini büyüklüklerine göre sıralamak için kullanılmaktadır. Burada esas amaç, DNA moleküllerinin bir kısmının negatif, bir kısmının ise pozitif elektrik yüküyle yükleyip, elektriksel alanda hareket ettirmektir. Elektriksel alanda hareket eden moleküllerden, boyu kısa olanlar daha hızlı ilerleyerek, diğer moleküllerin önüne geçerler ve bu şekilde DNA dizileri sıralanmış olur. Elektriksel alanda, moleküllerin hareketi rastgele olmayıp, negatif yüklü moleküller pozitif kutba doğru hareket ederken, pozitif yüklü moleküller negatif kutba doğru hareket ederler. Aynı yükle yüklü moleküller, aynı kutba hareket ederler. DNA tüpleri içerisinde hareket eden, DNA dizilerinin boyutunu ve büyüklüğünü göstermek için, ultraviyole ışınları ve renklendirme işlemi kullanılır [32]. Aynı zamanda, bu aşamalarda fotoğraf çekimi de yapılabilmektedir. Şekil 3.14'de gösterilen gel electrophoresis metodu, çok hassas ve güvenlidir.



DNA dizileri elektriksel alanda (-) kutuptan (+) kutba doğru hareket ederler. Küçük olan diziler daha önce (+) kutba ulaşır. Bu şekilde diziler boyutlarına göre sıralanır. (Gel Electrophoresis Yöntemi)

Şekil 3.14. DNA dizilerinin elektriksel alanda hareketi ve sıralanması

Gel electrophoresis içerisinde, DNA dizilerinin uzun olanları üst kısımda, kısa olanları ise alt kısımda kümelenmektedir. Bu kümelenmelerde, DNA dizilerinin yoğun olduğu kısımlar çok koyu renkli, daha az yoğun olduğu yerler ise açık renkli olarak görülür. Şekil 3.15'te, gel electrophoresis yöntemiyle çekilen bir fotoğraf gösterilmektedir [66].

Şekil 3.15. Gel electrophoresis fotoğrafı

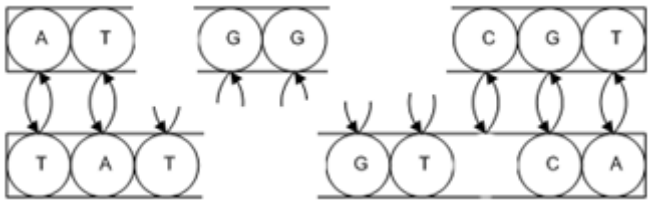
3.3.5. DNA Dizilerinin Çoğaltılması

DNA dizilerinin çoğaltılması için birçok yöntem vardır. Burada, sentez ve onarım yöntemleri birlikte kullanılır. Birden fazla metodun kullanıldığı bu işleme, PCR (Polymerase Chain Reaction) denilmektedir. Bu yöntemle, DNA dizilerinin onarımı ve kopyalanarak çoğaltılması sağlanmaktadır [33]. DNA dizilerinin onarımını gerçekleştirmek için, ligase ismi verilen enzimler kullanılır. Bu enzimler, DNA dizileri içerisinde bulunan bozulmuş alanları bularak onarırlar.

PCR, DNA dizilerinin kopya sayısını hızla artırmanın en iyi metodudur. Bu metot, DNA hesaplamadaki bazı problemlerin çözümünde kullanılır. DNA hesaplamadaki problemlerden biri, milyonlarca DNA dizisi içerisinden, problemin gerçek çözümü olan DNA dizisini bulup ayırmaktır. Bu yolla çözüm yapılırken, aranılan DNA dizisinin başlama ve bitim noktaları ve çözüm kümesi kodlanarak sisteme verilir. Sistem içerisinde, bu DNA kalıplarının milyonlarca kopyası oluşturularak, çözüm kümesinin daha hızlı bir şekilde bulunması sağlanır. Bu yöntemle, n tane işlem basamağı yapılarak, 2^n tane DNA parçası oluşturulur.

3.3.6. DNA Dizilerinden Parça Çıkarma Enzimleri

Bu enzimler, DNA dizilerini belirli amaca yönelik ve istenilen bölümlerinden kesilmesini sağlayan, biyokimyasal işlemleri gerçekleştirirler [66]. Bu işlemler için birçok enzim vardır. Bunlardan biri, Sau3AI enzimidir ve bu enzim sadece bazı DNA parçalarını kesebilir. GATC dizi parçası, Sau3AI enzimi tarafından kolayca tanınarak, o noktadan kesme işlemini gerçekleştirir. Şekil 3.16'da, DNA dizilerinin enzimler tarafından kesilmiş hali gösterilmiştir.



Şekil 3.16. Enzimler kullanılarak DNA dizilerinin kesilmesi

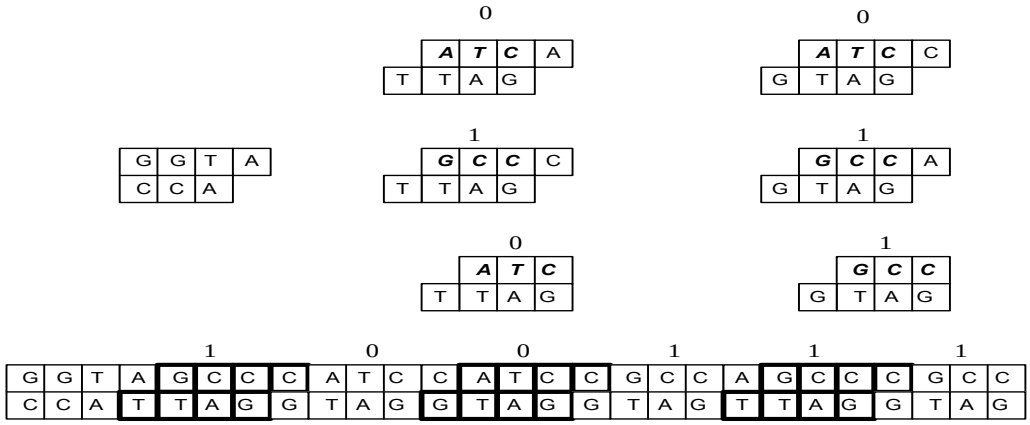
3.4. DNA Hesaplama için Geliştirilmiş Modeller

Bugüne dek, birkaç DNA hesaplama modeli geliştirilmiş olup, bu modeller kullanılarak bazı algoritmalar yazılmıştır. Bütün bu durumlara rağmen, geliştirilen modeller yeterli verimliliği sağlayamamıştır [33]. Şimdi bu modellerden kısaca bahsedelim.

3.4.1. DNA Dizilerinin Bireysel Topluluk Yöntemi ile Modellenmesi

Bireysel topluluk modeli (BTM), büyük boyutlarda moleküler yapıların inşası için geliştirilmiştir. Doğal hayatta bulunan atomların bir araya gelerek molekülleri oluşturması, moleküllerin yıldızları, gezegenleri ve galaksiyi oluşturması bireysel topluluk modeli için örnek olarak verilebilir [33]. Kimyasal moleküllerin sentezinde ve karmaşık moleküllerin inşasında BTM sıklıkla kullanılmaktadır. Bu modelde, tekli DNA dizilerinin sentezi kolayca yapılmakta ve DNA dizilerinin çaprazlanması işlemi kullanılmaktadır. Çaprazlama işlemi sayesinde, tekli DNA dizi parçaları birbirlerinin sonlarına bağlanarak, uzun DNA dizilerini oluşturmaktadır. Bu uzun yapıya bireysel topluluk denilmektedir. Self-Assembly (SA), bir kural yazma yöntemidir. DNA dizilerinin ne şekilde birleşeceği, yazılacak

kurallarla belirlenip, sentez işlemi gerçekleştirilir. Şekil 3.17'de, örnek bir gramer verilmektedir. Bu gramer kullanılarak, 100111 ikili bit dizisi DNA molekülleri ile kodlanmakta ve birleştirilerek SA yapısı oluşturulmaktadır. Örnek olarak verilen modelde, ATC dizisi 0 bit değerini, GCC dizisi ise 1 bit değerini ifade etmektedir. Bu yapılar, değişik şekillerde birleştirilerek DNA dizilimleri yapılır.



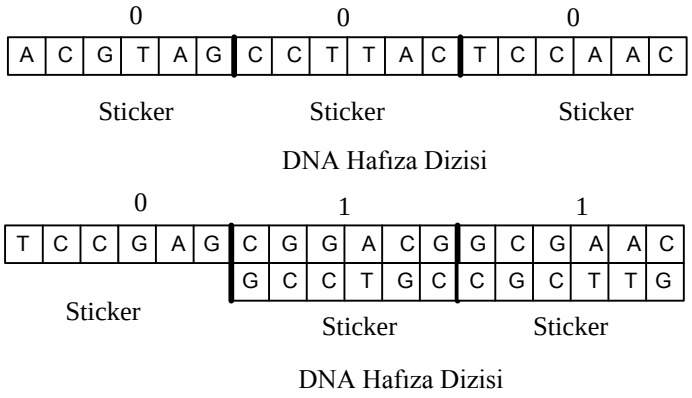
Şekil 3.17. Bireysel topluluk modeli ve örnek kodlama

SA modeli uygulanırken, öncelikle oligonükleotit'ler karıştırılarak, uzun DNA dizileri elde edilir. Bu DNA dizileri ile popüler SA yapısı oluşturulur. DNA dizilerini ayırma yöntemlerinden herhangi biri kullanılarak, çözüm kümesi bulunmaktadır. DNA SA modelinin en büyük dezavantajı, çok yavaş çalışması ve sıcaklık artışının fazla olmasıdır.

3.4.2. DNA Dizilerinin Etiketlendirme ile Modellenmesi

Etiket esaslı model, uzun DNA dizileri oluşturarak, karmaşık yapıların tasarlanmasını sağlayan ve bu şekilde hesaplama yapan, bir metot olarak geliştirilmiştir. Bu modelde, tek dizilimli DNA dizi parçaları 0 bit değeri ile, çift dizilimli DNA dizileri 1 bit değeri ile ifade edilmektedir [33]. DNA dizilerinin maksimum uzunluğunu L ile gösterirsek, N tane dizi bölgesi ve bu bölge içerisinde M tane DNA bazı varsa, L>=N.M olmak zorundadır. Yani, bölgeler ile o bölgelerdeki baz sayısının çarpımı, maksimum DNA uzunluğuna eşit veya küçük olmalıdır. Etiket temelli modele göre, bölgeler bir bit değeri ile gösterilir. Her bir N tane bölgeye, etiket (sticker) denir. DNA hafızası, M tane baz uzunluğuna sahip, N tane hafıza bölgesinden oluşmaktadır. Bu M tane baz'ın her biri, aynı zamanda bir tane

tümleyene sahiptir. Eğer etiket parçaları uygun bölgeye bağlanırsa, bu bölgelerin bit'i ON durumuna gelir. Eğer etiket ile eşleşecek hiç bölge yoksa, o bölgelerin bitleri OFF durumuna gelir. Şekil 3.18 ve Şekil 3.19'da, etiket temelli modelin kullanımı gösterilmektedir.

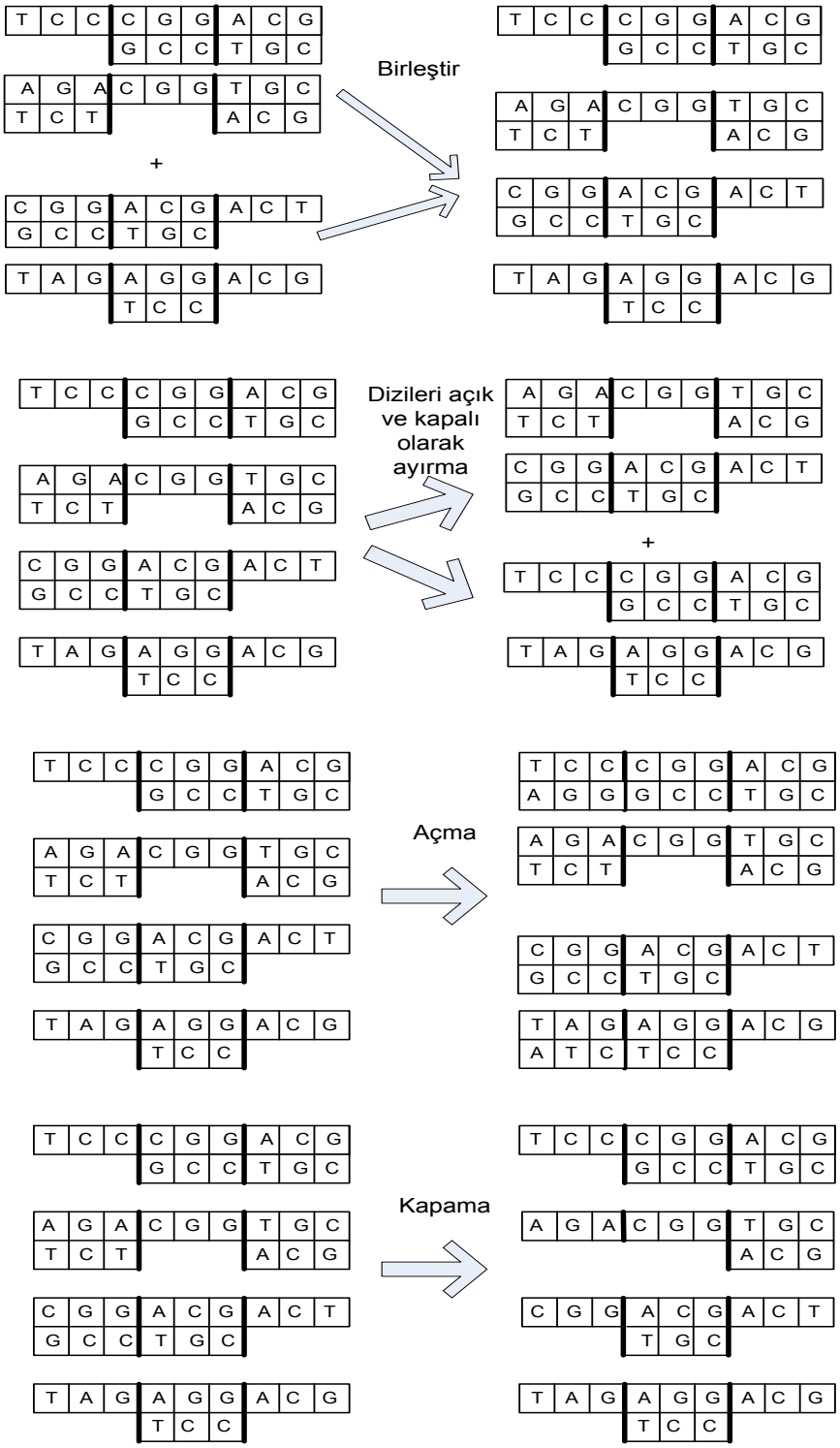


Şekil 3.18. Etiket temelli model

Etiket dizisi, bir bit veri ile ifade edilmektedir. Bit değerleri 0 veya 1 olmaktadır. Uygun bölgeyle eşleşen etiket dizileri 1 ile diğer etiket dizileri ise 0 ile gösterilir. Bu diziler çoğaltılarak, karmaşık dizi kümeleri oluşturulmaktadır. Her bir karmaşık hafıza kümesi, bir tüp olarak isimlendirilir ve hesaplama yapılırken kullanılır. Etiket esaslı model; birleştirme, ayırma, kümeleme (açma) ve temizleme (kapama) işlem birimlerine sahiptir. *Birleştirme* aşamasında, iki karmaşık DNA dizi kümesi birleştirilerek, yeni bir küme oluşturulmaktadır. Böylece, mümkün olan hafıza kümelerinin tamamı elde edilmektedir. *Ayırma* aşamasında, DNA kümeleri bölünerek, iki yeni küme oluşturulmaktadır. Bu yeni kümelerden bir kısmı 1 durumunda bit içerir, diğer bir kısmı ise 0 durumunda bit içerir. Bu işlem sonunda, açık olan etiketler bir tüpte, kapalı olan etiketler başka bir tüpte toplanır. *Kümeleme* aşamasında, DNA hafıza dizisi içerisinden belirlenen kapalı durumdaki (0 bit) bazı etiketler, açılarak 1 durumuna getirilir. *Temizleme* aşaması, karmaşık DNA hafızasından kapalı olan bitlerin çıkarılması olayıdır. Bu işlem sonucunda, hafızada kümelenmeyen dizilim kalmaz. Bütün bölgeler 1 değerini alırlar.

Bu modelin avantajı, ilk çözüm kümesinin kolay oluşturulmasıdır. Çözüm kümesindeki bütün diziler, aynı olmak zorundadır. İlk çözüm kümesinin sentezlenmesi, standart işlemlere göre çok hızlı ve ucuz olmaktadır. Bu modelin dezavantajı ise, bağlanan uzun

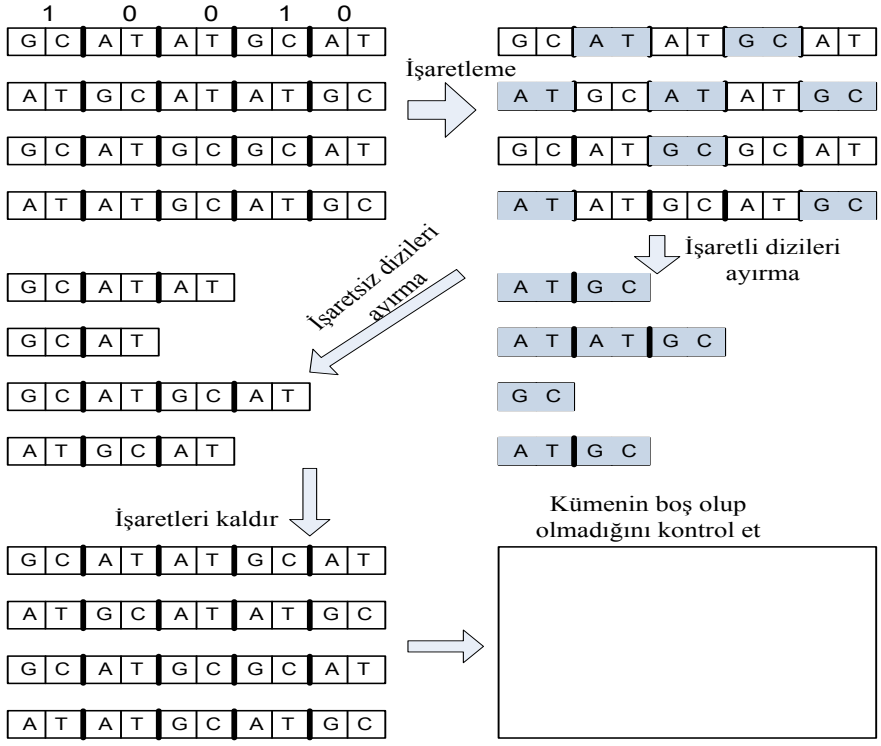
DNA dizilerinden dolayı, işlemlerin kesilme veya durma riskinin büyük olmasıdır. Aynı zamanda, clear (temizleme) aşaması çok zor gerçekleşmektedir.



Şekil 3.19. Etiket temelli modelde birleştirme, ayırma, açma ve kapama işlemleri

3.4.3. DNA Dizilerinin Yüzeyselleşme ile Modellenmesi

Bu modelin kullanım amacı, DNA dizilerini her hangi bir özel kap içerisinde, hareketsizleştirme işlemine tabii tutmaktır [33]. Özel kaptaki bu diziler, daha sonra melezleşme ve ayırma gibi DNA işlemlerinde kullanılır. Bu modelle ayrıca, kayıp DNA molekülleri kolayca üretilir. DNA bazlarının sentezi, protein dizilerinin analizi ve daha birçok kimyasal işlem, kolayca gerçekleştirilir. L uzunluğunda 0 ve 1 ikili bit dizisine sahip bir K kümesi olsun. K={1, 1, 0, 0, 0, 0, 1, 1, 1, 1} kümesinin uzunluğu L=10 olur. Küme içerisindeki her bit, DNA dizileri kullanılarak kodlanır. Örneğin GC-1, AT-0 olarak seçilir. Yüzey esaslı model; işaretleme, işaretli dizileri çıkarma, işaretsiz dizileri çıkarma, işaretli dizilerin işaretlerini kaldırma ve kümenin boş olup olmadığını test etme işlem basamaklarına sahiptir. *İşaretleme* adımında, küme içerisinden belirlenen elemanlar işaretlenir. *İkinci adımda,* işaretli diziler kümeden çıkarılır. *Üçüncü adımda,* işaretsiz diziler kümeden çıkarılır. *Dördüncü adımda,* işaretli dizilerin işaretleri kaldırılır. *Beşinci adımda,* kümenin boş olup olmadığı kontrol edilir. Küme boş ise çözüm yoktur. Küme boş değilse, bulunan değerler çözüm olarak alınır. Şekil 3.20'de verilen algoritmada, küme boş olduğu için çözüm yoktur.



Şekil 3.20. Yüzey esaslı modelde algoritmik adımların gösterimi

Bu modelin diğerlerinden en önemli farkı, DNA moleküllerinin sabitleştirilmesidir. Ayrıca, ilk çözüm kümesinin oluşturulması çok kolaydır. Aynı zamanda, çözüm alanının oluşturulması çok hızlı ve maliyeti az olmaktadır. Bu modelin dezavantajı, sadece iki boyutlu problemlerde kullanılabilmesidir. Eğer daha fazla boyuta uygulanmak istenirse, yüzey yoğunluğunu, yüzey alanını ve kimyasal alanı genişletmek zorunludur. Şekil 3.20'de, yüzey esaslı modelin kullanımı verilmektedir.

3.5. Bölüm Değerlendirmesi

Literatürde var olan sayısal DNA hesaplama algoritmalarında mutasyon uygulanırken, DNA dizilerinin tersten yazılması ve kaydırılması işlemleri uygulanmıştır. Rastlantısal olarak oluşturulan popülasyon elemanları, aynı dizilerden oluşursa, tersten yazma ve kaydırma durumunda, popülasyon değişmeden devam etmektedir. Bu tez çalışmasında mutasyon uygulanırken, önce belirli oranda popülasyon elemanı silinip, yerine tamamen yeni diziler eklenmekte ve popülasyonun yenilenmesi sağlanmaktadır.

Yapılan uygulamada, literatürde var olan kompleks algoritmalar yerine, DNA hesaplama algoritması, daha basitleştirilerek algoritmanın uygulanabilirliği artırılmış ve tamamlanma süresi kısaltılmıştır.

4. NP PROBLEMLERDE DNA HESAPLAMANIN KULLANILMASI

Bugüne kadar karşılaşılan problemleri, zorluk ve karmaşıklık derecesine göre, 3 gruba ayırmak mümkündür: Çözümü kolay olan, çözümü zor olan ve çözümü olmayan problemler. Çözümü kolay olan problemler Polinomal (P), çözümü zor olan problemler ise NP olarak kabul edilmektedir [40, 71]. Bu tür problemlerin kesin çözümü olmadığı için, yapılan çalışmalarda, yaklaşık çözüm noktasına ulaşılmaya çalışılmaktadır. NP birçok problem olup; gezgin satıcı, sırt çantası, maksimum akış, mantıksal doğruluğu sağlama, bilgisayar ağlarında trafiğin yönetilmesi, çok sınıflı ve çok öğretmenli okullarda haftalık ders programının hazırlanması ve grafiklerin renklendirilmesi, bunlardan sadece birkaçıdır. NP problemlerin çözümünde; aritmetik yöntemler, sayısal analiz yöntemi ve optimizasyon algoritmalarından yararlanılmaktadır. Bu problemlerde, değişken sayısı arttıkça, sayısal analiz ve aritmetik yöntemlerin kullanımı zor olmakta ve çözümün elde edilmesi çok uzun zaman almaktadır [15]. Yapılan bilimsel çalışmalarda, bu tür problemlerin çözümü için, optimizasyon algoritmaları geliştirilmiştir. Geliştirilen bu algoritmalar, belirli bir çözüm alanı içerisinde, hızlı ve etkili arama gerçekleştirebilme özelliğine sahiptirler. Ayrıca, karmaşık ve zor matematiksel denklemlere gerek duymadan, sadece uygunluk fonksiyonunun belirlenmesi ile sonuca ulaşmaktadırlar.

DNA hesaplama, bir optimizasyon algoritması olup, bir çok NP problemin çözümünde kullanılmaktadır. Özellikle, gezgin satıcı, sırt çantası ve grafiklerin renklendirilmesi problemlerinde çok fazla uygulama mevcuttur [12, 15, 24, 36]. DNA hesaplama, çözelti ortamında yapılan uygulamalarda, büyük miktarda veri saklama kapasitesine ve paralel işlem yeteneğine sahiptir. Bu özelliği ile, değişken sayısı çok olan ve çözümü uzun süren NP problemler için, uygulanabilen en iyi algoritmalardan biridir. Adleman, DNA moleküllerinin bu önemli özelliklerini kullanarak, gezgin satıcı probleminin optimizasyonunu gerçekleştirmiştir [2]. Yapılan bu çalışma ile DNA hesaplamanın, NP problemlerin çözümünde, etkili bir yöntem olduğu keşfedilmiş ve takip eden yıllarda, birçok bilimsel çalışma gerçekleştirilmiştir. Bu bilimsel çalışmalarda, DNA hesaplama, farklı NP problemlere uygulanarak, çok iyi sonuçların bulunabileceği gösterilmiştir.

4.1. NP Problemler

Polinomal olmayan problemler genellikle karmaşık ve parametre sayısı çok fazla olduğu için elektronik bilgisayarlar ile çözümleri zordur. Çünkü elektronik bilgisayarların veri saklama kapasitesi, hızı ve problem çözme yeteneği sınırlıdır. Özellikle parametre sayısı arttıkça, problemin çözümü zorlaşmaktadır [32]. Bu sınırlamaları en aza indirmek ve daha etkili problem çözümü gerçekleştirmek için, optimizasyon algoritmaları geliştirilmiştir. Optimizasyon problemlerinin çözümünde kullanılan bu algoritmalardan biride, DNA hesaplamadır [40]. DNA hesaplama, paralel işlem yapma ve yüksek miktarda veri saklama yeteneği olan DNA moleküllerini kullanmaktadır. Bu güne kadar yapılan moleküler çalışmaların çoğu, biyoloji laboratuarlarında gerçekleştirilmiştir. Fakat laboratuar ortamında yapılan bu uygulamaların, pratik olarak kullanımı büyük bir maliyet gerektirmekte ve uygulanması zor olmaktadır. Halen devam eden bilimsel çalışmalar ile, DNA moleküllerinden oluşan, DNA bilgisayarların yapımı amaçlanmaktadır. DNA bilgisayarların kullanımı yaygınlaştığında, işlem yapma hızı ve veri saklama kapasitesinde hızla ilerleme sağlanacaktır.

0–1 programlama problemleri, polinomal olmayan denklemler kategorisinde yer almaktadır [8, 16]. Bu problemlerin çözümü, çok karmaşık ve zor olduğu için, sonuca ulaşmayı kolaylaştıracak iyi bir algoritma yazılarak, problem tüm yönleriyle açıklanmalıdır. Bu konuda, birçok algoritma yazılmasına rağmen, yeterli derecede sonuca götürecek bir ilerleme henüz olmamıştır. NP grubundaki bu problemlerin çözümüne yönelik, DNA hesaplamayı kullanarak biyoloji laboratuarında somut değerler bulan, ilk bilim adamı Adleman olmuştur [2]. Adleman, 7 şehirli bir gezgin satıcı problemini, DNA moleküllerini kullanarak kodlamış ve biyokimyasal tepkimelerle sonuç elde ederek, bulduğu değerleri, elektroskobik floresan üzerine yazdırmayı başarmıştır.

0-1 programlama problemini şu şekilde tanımlayabiliriz. Bu problemi oluşturan p tane değişken ve k tane denklem olsun [72]. Her bir değişken 0 veya 1 değeri alabilir. Sıfır, seçilen bir ise, seçilmeyen değişkenleri ifade etmektedir. Denklem 4.1'de, örnek bir optimizasyon problemi, Denklem 4.2, 4.3, 4.4 ve 4.5'te ise, bu problemin sınırlayıcıları verilmektedir.

$$\text{Min}(y)= a_1x_1 + a_2x_2 + a_3x_3 + \ldots.. + a_nx_n \qquad (4.1)$$

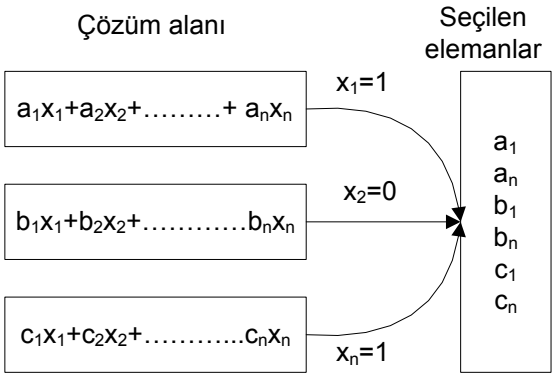
$$b_{11}x_1 + b_{12}x_2 + \ldots.. + b_{1p}x_p <= c_1 \quad (4.2)$$

$$b_{21}x_1 + b_{22}x_2 + \ldots.. + b_{2p}x_p <= c_2 \quad (4.3)$$

$$b_{k1}x_1 + b_{k2}x_2 + \ldots.. + b_{kp}x_p <= c_k \quad (4.4)$$

$$x_1, x_2, x_3 \ldots.. x_p = 0,1 \quad (4.5)$$

Yukarıda denklemleri verilen 0–1 programlama problemlerinin optimizasyonu, literatürde sıklıkla kullanılmaktadır [12]. Bu konuda, birçok algoritma yazılmasına rağmen, henüz tatmin edici derecede çözüm üretilememiştir. Şekil 4.1'de, örnek bir 0-1 planlama problemine ait denklem sistemi verilmektedir. Burada, x değişkeninin alacağı değere göre, çözüm için seçilecek elemanlar belirlenmektedir. X değişkeni, 0 veya 1 değerini almaktadır.

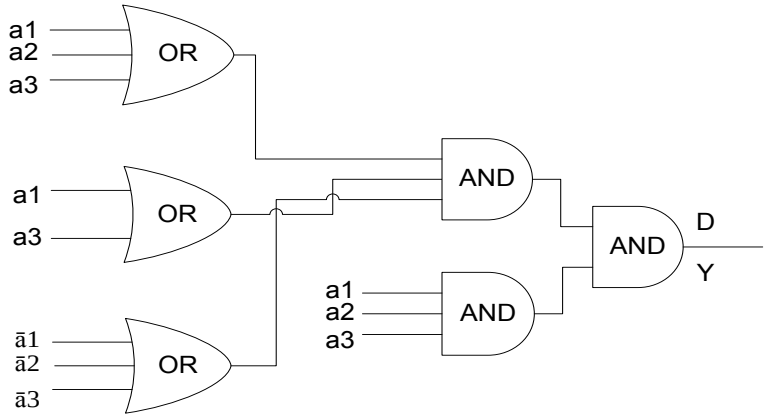


Şekil 4.1. Örnek bir 0/1 problemin gösterimi

Mantıksal doğruluğu sağlama problemleri, literatürde satisfiability (SAT) olarak kullanılmakta olup, NP-zor problem sınıfında yer almaktadır [9, 11]. SAT problemlerinde, değişken sayısı 3'ten fazla olunca, 3SAT olarak adlandırılmaktadır. Bu tür problemler, karmaşık ve zor bir yapıya sahip olup, uzun yıllar bilimsel literatürde araştırma konusu olmuştur. 3SAT problemlerini, şu şekilde tanımlayabiliriz. k tane değişken m_1, m_2, $m_3 \ldots m_k$ verilsin. Her değişken, sadece doğru veya yanlış değerlerinden birini alabilir. Bu tür problemlerin gösterimi, Denklem 4.6'da verilmektedir.

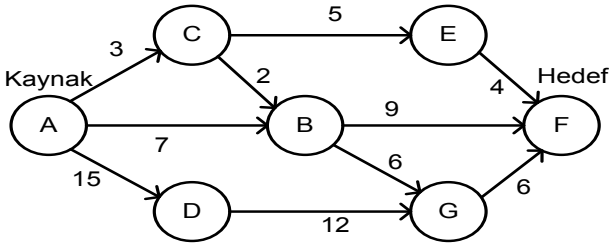
$$(a_1 \text{ V } a_2 \text{ V } a_3)(a_1 \text{ V } a_3)(a_2 \Lambda a_1 \Lambda a_3 \ )(\ \bar{a}_1 \text{V } \bar{a}_2 \text{ V } \bar{a}_3\ ) \quad (4.6)$$

Denklem 4.6'da verilen V simgesi *veya* bağlacı, Λ simgesi ise *ve* bağlacını ifade etmekte olup, mantıksal değişkenler arasındaki ilişkileri tanımlamak için kullanılırlar. Şekil 4.2'de, 3SAT problemleri şematik olarak gösterilmektedir. Bu problemlerinin optimizasyonunda amaçlanan, mantıksal denklemlerin doğruluğunun sağlanmasıdır.



Şekil 4.2. SAT problemlerinin gösterimi

Maksimum akış problemleri, NP olarak kabul edilmiş olup, DNA hesaplama ile çözümü üzerine çalışılan, bir başka optimizasyon problemidir. Şebeke problemlerini içeren bu sistemler, m düğümlü, n kenarlı bir Ş şebekesinde, kaynaktan hedefe doğru akan nesne miktarının hesaplanmasını içermektedirler. Düğüm ve kenar sayısı arttıkça, problemlerin çözümü zorlaşmaktadır. Şekil 4.3'te, örnek bir maksimum akış problemi verilmekte olup, bu problem, 7 düğüm ve 10 kenardan oluşmaktadır [16].



Şekil 4.3. Örnek bir maksimum akış problemi

Maksimum akış problemleri; daha çok taşımacılıkta, iletişimde, bilgisayar ağlarının yerleştirilmesi ve yönetilmesinde, trafik lambalarının optimizasyonu ve elektrik dağıtım şebekelerinin yerleştirilmesi işlemlerinde kullanılmaktadır.

Tez çalışmasının bu bölümünde, DNA hesaplama algoritmasının, polinomal olmayan iki probleme uygulanması yapılmıştır. Örnek olarak seçilen bir gezgin satıcı ve sırt çantası probleminin, optimum çözümü gerçekleştirilmiştir. Problemleri minimum ve maksimum yapan değerler, DNA hesaplama algoritması kullanılarak bulunmuştur.

4.2. Gezgin Satıcı Probleminin DNA Hesaplama Çözümü

Gezgin satıcı problemi, NP-zor problemlerden olup, optimizasyon algoritmalarının tamamına yakını için, örnek uygulama olarak kullanılmıştır. DNA hesaplama algoritması ile yapılan ilk uygulamada, gezgin satıcı problemidir [2]. Adleman, yaptığı uygulamada, 7 şehirli bir gezgin satıcı problemini, biyoloji laboratuarında, DNA moleküllerini kullanarak çözmüştür. Şehir sayısı ve şehirleri bağlayan yol sayısı arttıkça, bu problemin çözümü zorlaşmakta ve çözüm süresi uzamaktadır. Gezgin Satıcı Problemini, şu şekilde tanımlayabiliriz. N tane şehir bulunmaktadır. Gezgin Satıcı olarak isimlendirdiğimiz bir kişi, bu şehirlerden her birine *en az* ve *sadece bir kez* uğramak zorundadır. Ayrıca, bizim istediğimiz bir başlangıç noktasından başlayıp, yine bizim istediğimiz bir bitim noktasına ulaşmak durumundadır. Bu problemde amacımız, bütün bu şartların gerçekleştiği, minimum mesafeli yolu bulmaktır. Gezgin satıcı probleminde, yol mesafeleri ve şehir sayısı arttıkça, elektronik bilgisayarlar ile problemin çözümü zorlaşmaktadır. Bu durumda, DNA dizilerini kullanan bilgisayarlar etkili olmakta, zaman ve performans açısından başarı sağlamaktadır. Çünkü DNA bilgisayarlar, paralel işlem yöntemini kullanmakta ve problemi kısa sürede çözmektedir. Elektronik bilgisayarlar, seri işlem yöntemini kullandıkları için, paralel işlem gerektiren problemleri, DNA bilgisayarlara göre daha uzun sürede ve daha zor çözmektedirler.

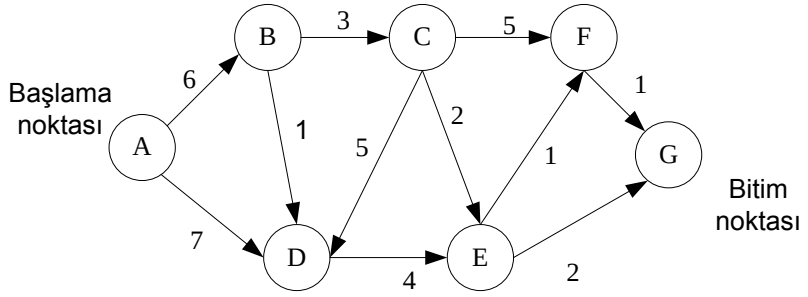
Bu bölümde, Şekil 3.4'te örnek olarak seçilen gezgin satıcı problemi için, DNA hesaplama algoritması kullanılarak, optimum çözüm bulunmaktadır. Problemin çözümü için, matlab 7.0 ile m-file yazılarak, DNA hesaplama algoritması programlanmaktadır. Simülasyon için kullanılan parametre değerleri, Tablo 4.1'de verilmektedir. Bu parametrelere ait değerler, uygulayıcı tarafından belirlenir. Problemin türüne ve zorluk derecesine göre, popülasyon büyüklüğü ve maksimum işlem sayısı artırılıp azaltılabilir.

Enzim ve virüs oranı, her bir iterasyon için kontrol edilerek, ideal değerler belirlenebilir. Bu oranlar, gereğinden fazla artırıldığında veya çok sınırlı düzeyde uygulandığında, çözüm kümesi için uygun olan değerlerin kaybına veya yerel optimum noktalara odaklanmaya neden olmaktadır.

Tablo 4.1. Gezgin Satıcı Problemi için kullanılan DNA hesaplama parametreleri

Parametre	Değeri
Şehir Sayısı	7
Populasyon Büyüklüğü	20
Maksimum İşlem Sayısı	100
Çaprazlama Oranı	%100
Enzim Oranı	0.3
Virüs Oranı	0.3

Örnek problem, 7 şehirden oluşmakta ve her şehir birbirine yollarla bağlanmaktadır. Problem için, A şehri başlama noktası, G şehri ise bitim noktası olarak seçilmiştir. Bu optimizasyon probleminde amaç, A şehri ile G şehri arasındaki bütün şehirlere bir kez uğrayarak, gerçekleşecek en kısa mesafeli yolu bulmaktır. Aşağıda, m-file için kullanılan DNA hesaplama algoritmasının, algoritmik adımları ayrıntılı olarak açıklanmaktadır.



Şekil 4.4. Örnek bir gezgin satıcı problemi

Problemin DNA hesaplama algoritması şöyledir:

Adım 1: Şehir sayısı 7 olarak seçilmiş olup, şehirlerarasındaki yollar ve uzunlukları, DNA molekülleri ile kodlanmakta ve ilk popülasyon rastlantısal olarak oluşturulmaktadır.

Kodlama

Şehir ve mesafelerin kodlaması için, DNA dizileri kullanılmaktadır. Her bir şehir ve mesafe, 3'lü DNA dizisi ile gösterilmekte ve A-0, G-1, C-2 ve T-3 değerleri kullanılarak sayısal değerlere çevrilmektedir. Örneğin, A şehri ile B şehri arasındaki mesafe 6 birim olduğundan, AGC ile gösterilmektedir. A şehri 1. şehir olduğu için, AAG ile kodlanmaktadır. Şekil 4.4'te verilen problem için, şehirlerin ve yolların DNA ile kodlanması, Tablo 4.2 ve Tablo 4.3'te ayrıntılı olarak verilmektedir. Ayrıca, Tablo 4.4'te, Tablo 4.3'te verilen DNA dizilerinin, sayısal olarak karşılıkları verilmektedir.

Tablo 4.2. Şehirlerin DNA dizileri ile gösterimi

Şehirler	DNA ile gösterimi	4'lü sayı sistemindeki karşılığı	10'lu sayı sistemindeki karşılığı
A	AAG	001	1
B	AAC	002	2
C	AAT	003	3
D	AGA	010	4
E	AGG	011	5
F	AGC	012	6
G	AGT	013	7

Tablo 4.3. Şehirlerarasındaki mesafelerin DNA dizileri ile gösterimi

Şehirler	1	2	3	4	5	6	7
1	AAA	AGC	AAC	AGT	ACC	ATC	GAA
2	AGC	AAA	AAT	AAG	ACA	ACG	ACC
3	AAC	AAT	AAA	AGG	AAC	AGG	AGC
4	AGT	AAG	AGG	AAA	AGA	AAT	AGG
5	ACC	ACA	AAC	AGA	AAA	AAG	AAC
6	ATC	ACG	AGG	AAT	AAG	AAA	AAG
7	GAA	ACC	AGC	AGG	AAC	AAG	AAA

Tablo 4.4. Şehirlerarasındaki mesafelerin gerçek değerleri

Şehirler	1	2	3	4	5	6	7
1	0	6	2	7	10	14	16
2	6	0	3	1	8	9	10
3	2	3	0	5	2	5	6
4	7	1	5	0	4	3	5
5	10	8	2	4	0	1	2
6	14	9	5	3	1	0	1
7	16	10	6	5	2	1	0

Tablo 4.5. Birinci adım için rastlantısal olarak yolların oluşturulması

Yollar	Seçilen şehirler	DNA dizileri ile gösterimi
Y1	1-2-3-4	AGC AAT AGG
Y2	5-6-7	AGG AGC AGT
Y3	3-4-5-6-7	AAT AGA AGG AGC AGT
Y4	1-2	AAG AAC
Y5	1-3-5-7	AAG AAT AGG AGT
Y6	2-4-5-6	AAC AGA AGG AGC
Y7	1-2-3-4-5-6	AAG AAC AAT AGA AGG AGC
Y8	1-2-3-4-5-6-7	AAG AAC AAT AGA AGG AGC AGT
Y9	1-2-3-4-5-6-1-7	AAG AAC AAT AGA AGG AGC AAG AGT
Y10	1-2-3-7	AGC AAT AGC
Y11	1-3-2-4-6-5-7	AAC AAT AAG AAT AAG AAC
Y12	1-3-2-4-5-6-7	AAC AAT AAG AGA AAG AAG

Tablo 4.5'te verildiği gibi, ilk popülasyon şehirlerin farklı şekillerde dizilimlerinden oluşturulmaktadır. Bu dizilimler, DNA bazları ile oluşturulup, sonra sayısal verilere çevrilmektedir. Bu adımda, bilgisayarda gerçekleşen işlemin belirli bir kısmı gösterilerek, çözümün anlaşılması sağlanmaktadır.

Adım 2: Yollar için oluşturulan popülasyonun her bir elemanı için, uygunluk değerleri hesaplanmaktadır. Popülasyonda bulunan rastlantısal dizilimler için, Denklem 4.1'de verilen uygunluk fonksiyonu kullanılarak, uygunluk değerleri elde edilmektedir. Gezgin satıcı probleminde, t tane şehir olsun. Bu durumda, şehirlerin kümesi Ş={ $ş_1$, $ş_2$, $ş_3$,….., $ş_t$} şeklinde olur. Şehirlerarasındaki mesafeleri m($ş_i$, $ş_j$) ile gösterirsek, uygunluk fonksiyonu Denklem 4.7'de olduğu gibi elde edilir.

$$\min\left((\sum_{i=1}^{t-1} m(ş_i, ş_{i+1})) + m(ş_t, ş_1) \right) \tag{4.7}$$

Tablo 4.5'te oluşturulan yolların uygunluk değerleri, Tablo 4.6'da verilmektedir. Uygunluk değerleri, Tablo 4.4'teki verilerin, Denklem 4.7'de yerine konulmasıyla elde edilmektedir.

Tablo 4.6. İkinci adım için uygunluk değerlerinin hesaplanması

Yollar	Seçilen şehirler	DNA dizileri ile gösterimi	Uygunluk değerleri
Y1	1-2-3-4	AGC AAT AGG	6+3+5=14
Y2	5-6-7	AGG AGC AGT	1+1=2
Y3	3-4-5-6-7	AAT AGA AGG AGC AGT	5+4+1+1=11
Y4	1-2	AAG AAC	6
Y5	1-3-5-7	AAG AAT AGG AGT	2+2+2=6

Tablo 4.6'nın devamı

Y6	2-4-5-6	AAC AGA AGG AGC	1+4+1=6
Y7	1-2-3-4-5-6	AAG AAC AAT AGA AGG AGC	6+3+5+4+1=19
Y8	1-2-3-4-5-6-7	AAG AAC AAT AGA AGG AGC AGT	6+3+5+4+1+1=20
Y9	1-2-3-4-5-6-1-7	AAG AAC AAT AGA AGG AGC AAG AGT	6+3+5+4+1+14+16=49
Y10	1-2-3-7	AGC AAT AGC	6+3+6=15
Y11	1-3-2-4-6-5-7	AAC AAT AAG AAT AAG AAC	2+3+1+3+1+2=12
Y12	1-3-2-4-5-6-7	AAC AAT AAG AGA AAG AAG	2+3+1+4+1+1=12

Adım 3: Problem için, bir başlama ve bitim noktası belirlenerek, oluşturulan yollardan bu koşula uymayanlar çıkarılmaktadır. Örnek problemde, A şehri başlama noktası G şehri ise bitim noktası olarak kabul edilmektedir. Bu durumda, Y5, Y8, Y9, Y10, Y11 ve Y12 yollarının dışında kalan elemanlar, çözüm kümesinden çıkarılmaktadır. Kalan yollar, Tablo 4.7'de verilmektedir.

Tablo 4.7. Başlama ve Bitim noktası kontrolü sonucu kalan yollar

Yollar	DNA dizileri ile gösterimi	Uygunluk değerleri
Y5	AAG AAT AGG AGT	2+2+2=6
Y8	AAG AAC AAT AGA AGG AGC AGT	6+3+5+4+1+1=20
Y9	AAG AAC AAT AGA AGG AGC AAG AGT	6+3+5+4+1+14+16=49
Y10	AGC AAT AGC	6+3+6=15
Y11	AAC AAT AAG AAT AAG AAC	2+3+1+3+1+2=12
Y12	AAC AAT AAG AGA AAG AAG	2+3+1+4+1+1=12

Adım 4: Problem için belirlenen şehirlerin tamamına uğramayan yollar, çözüm kümesinden çıkarılmaktadır. Örnek uygulamada, 7 şehir olduğu için, gezgin satıcı tüm şehirlere uğramak zorundadır. Yapılan işlem sonucunda, Y8, Y9, Y11 ve Y12 yolları çözüm kümesinde kalmaktadır. Kalan yollar Tablo 4.8'de verilmektedir.

Tablo 4.8. Tüm şehirlere uğrama kontrolü sonucu kalan yollar

Yollar	DNA dizileri ile gösterimi	Uygunluk değerleri
Y8	AAG AAC AAT AGA AGG AGC AGT	6+3+5+4+1+1=20
Y9	AAG AAC AAT AGA AGG AGC AAG AGT	6+3+5+4+1+14+16=49
Y11	AAC AAT AAG AAT AAG AAC	2+3+1+3+1+2=12
Y12	AAC AAT AAG AGA AAG AAG	2+3+1+4+1+1=12

Adım 5: Bir şehre birden fazla uğrayan yollar çözümden çıkarılıp, simülasyonun ilk iterasyonu sonucu elde edilen küme içerisinden, minimum mesafe ve en kısa yol

belirlenmektedir. Bu adım sonucu kalan yollar, Tablo 4.9'da verilmektedir. Y9 yolunda, A şehrine iki kez uğranıldığı için bu yol çözümden çıkarılır

Tablo 4.9. Şehirlere bir kez uğrama koşulu sonucu kalan yollar

Yollar	DNA dizileri ile gösterimi	Uygunluk değerleri
Y8	AAG AAC AAT AGA AGG AGC AGT	6+3+5+4+1+1=20
Y11	AAC AAT AAG AAT AAG AAC	2+3+1+3+1+2=12
Y12	AAC AAT AAG AGA AAG AAG	2+3+1+4+1+1=12

Adım 6: Yollar için oluşturulan popülasyona, çaprazlama uygulanarak, yeni popülasyon oluşturulmakta ve 2, 3, 4, 5 adımları tekrarlanıp, minimum mesafe ve en kısa yol yeniden belirlenmektedir.

Adım 7: Yollar için oluşturulan popülasyona, enzim ve virüs mutasyonu uygulanarak, yeni popülasyon oluşturulmakta ve 2, 3, 4, 5 adımları tekrarlanıp, minimum mesafe ve en kısa yol yeniden belirlenmektedir.

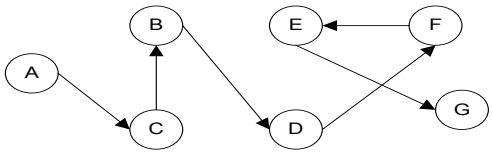
Adım 8. Maksimum işlem sayısına ulaşıldığında, bulunan küme içerisinden, minimum mesafe ve en kısa yol belirlenerek, problemin optimum çözümü elde edilmektedir.

Şekil 4.4'de verilen problemin, DNA algoritması ile çözümü sonucunda, aşağıdaki değerler elde edilmiştir.

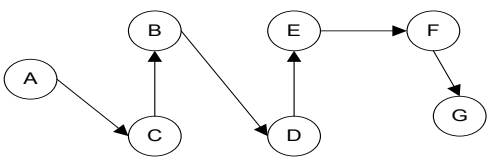
Bulunan en kısa yol uzunluğu, 12 birimdir. Uzunluğu 12 birim olan, 2 tane yol güzergâhı belirlenmiştir. Şekil 4.5 ve şekil 4.6'da, bulunan yol güzergahları gösterilmektedir.

1. yol güzergahı 1 → 3 → 2 → 4 → 6 → 5 → 7

2. yol güzergahı 1 → 3 → 2 → 4 → 5 → 6 → 7



Şekil 4.5. Bilgisayar simülasyonu sonucu bulunan 1. Yol güzergahı



Şekil 4.6. Bilgisayar simülasyonu sonucu bulunan 2. Yol güzergahı

4.3. Sırt Çantası Probleminin DNA Hesaplama Çözümü

Sırt çantası problemi şu şekilde tanımlanır. Elimizde, ağırlıkları ve kazançları farklı olan elemanlar bulunmaktadır. Bu elemanlar, kapasitesi sınırlı olan ve bilinen bir kutu veya benzeri bir araç içerisine yerleştirilecektir. Bu problemin çözümünde amaç, *maksimum kazanç sağlanacak şekilde* ve *kapasiteyi geçirmeyecek şekilde*, en fazla ağırlıkta eleman yerleştirmektir.

Bu tür problemler, 0/1 planlama problemi olarak tanımlanır [5, 71]. Burada, kazanç kümesi içerisine girebilen elemanlar 1 ile diğer elemanlar ise 0 ile işaretlenmekte ve hesaplama ona göre gerçekleşmektedir.

Sırt çantası problemi için, n tane eleman verilsin ve çantamızın kapasitesi c olsun. Burada, her bir i elemanı için kazanç k_i ve ağırlık a_i olsun. Toplam ağırlık, c değerini aşmamak üzere, toplam kazanç maksimum olmak zorundadır. Elemanların, çözüm kümesinin bir parçası olup olmadığını, d kümesi ile gösterirsek d={0, 1} olur. Burada, 0 çözüme dahil olmayan, 1 ise çözümün bir parçası olan elemanları göstermektedir. Uygulama için seçilen Sırt çantası problemi çözülürken, Denklem 4.8 ve Denklem 4.9 kullanılmıştır.

$$\sum_{i=1}^{n} a(i)*d(i) \le c \tag{4.8}$$

$$Maksimum\_Kazanc = \sum_{i=1}^{n} k(i)*d(i) \tag{4.9}$$

0/1 planlama problemlerinin çözümünde, birçok algoritma kullanılmıştır. Bu bölümde, Şekil 4.5'te verilen, örnek sırt çantası probleminin, DNA hesaplama algoritması kullanılarak, optimum çözümü gerçekleştirilmiştir. Problemin çözümü için matlab 7.0 ile m-file yazılarak DNA hesaplama algoritması programlanmaktadır. Şekil 4.7'de seçilen sırt çantası problemi, farklı ağırlık ve kazanç değerlerine sahip, 10 elemandan oluşmaktadır. Bu problemde amaç, kapasitesi 85 olan bir kutu içerisine, en fazla nesneyi yerleştirmektir. Yerleştirme yapılırken, yerleşen elemanların toplam ağırlığı, kapasite değerini geçmemelidir. DNA hesaplama için kullanılan parametre değerleri, Tablo 4.10'da verilmektedir.

Tablo 4.10. Sırt Çantası Problemi için kullanılan DNA hesaplama parametreleri

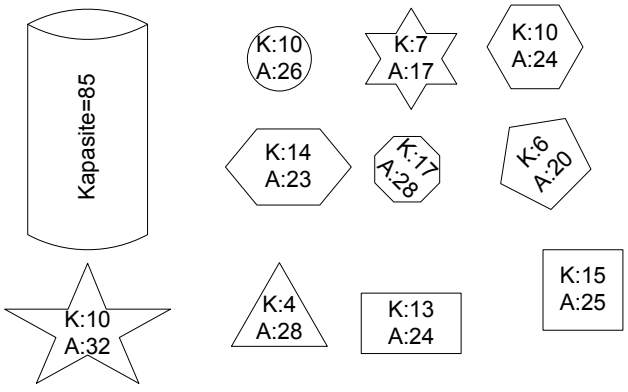
Parametre	Değeri
Eleman Sayısı	10
Popülasyon Büyüklüğü	150
Maksimum İşlem Sayısı	100
Kapasite	85
Mutasyon Oranı	0.3
Enzim Oranı	0.3
Virüs Oranı	0.3

Şekil 4.7'de verilen elemanlar ile ağırlık ve kazanç değerleri, liste şeklinde aşağıda verilmektedir.

E = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

K= { 10, 7, 10, 14, 17, 6, 13, 4, 10, 15}

A = {26, 17, 24, 23, 28, 20, 24, 28, 32, 25}, c=85.



Şekil 4.7. Örnek olarak kullanılan sırt çantası problemi

Aşağıda, m-file için kullanılan DNA hesaplama algoritmasının, algoritmik adımları ayrıntılı olarak açıklanmaktadır:

Adım 1: Eleman sayısı 10 olarak belirlenen problemin elemanlarına ait, ağırlık ve kazanç değerleri, Tablo 4.5'te verilmektedir. Kapasite değeri 85 olarak alınmıştır. Ağırlıklar ve kazançlar, DNA molekülleri kullanılarak kodlanmıştır.

Kodlama:

Elemanların ağırlık ve kazanç değerleri ile, bunların seçiminde kullanılacak popülasyon, DNA dizileri kullanılarak kodlanır. Her ağırlık ve kazanç, 3'lü DNA dizisi ile gösterilmektedir. Örneğin, 1. elemanın ağırlığı 25 olduğu için, bu değere karşılık gelen GCG DNA dizisi kullanılarak gösterilmektedir. Yine, 1. elemanın kazanç değeri 9 olduğu için, ACG dizisi ile kodlanmaktadır. Şekil 4.5'te verilen problem için, ağırlık ve kazanç değerlerinin DNA dizileri ile kodlanması, Tablo 4.11'de ayrıntılı olarak verilmektedir.

Tablo 4.11. Ağırlıkların ve Kazançların DNA dizileri ile gösterimi

Ağırlıklar	DNA ile gösterimi	4'lü karşılığı	Kazançlar	DNA ile gösterimi	4'lü karşılığı
26	GCC	122	10	ACC	022
17	GAG	101	7	AGT	013
24	GCA	120	10	ACC	022
23	GGT	113	14	ATC	032
28	GTA	130	17	GAG	101
20	GGA	110	6	AGC	012
24	GCA	120	13	ATG	031
28	GTA	130	4	AGA	010
32	CAA	200	10	ACC	022
25	GCG	121	15	ATT	033

Kutuya yerleşecek elemanları belirlemek için, DNA molekülleri kullanılarak ve rastlantısal olarak, ilk popülasyon oluşturulmakta ve 0-1 değerlerine çevrilmektedir. 0 seçilmeyecek elemanları, 1 ise seçilecek elemanları temsil etmektedir. Popülasyon oluşumunda, A-0, C - 0, G-1 ve T-1 olarak sayısal verilere çevrilmektedir. Popülasyon için, 10 baz uzunluğunda diziler kullanılmaktadır. İlk popülasyonun oluşumu, Tablo 4.12'de verilmektedir.

Tablo 4.12. Örnek sırt çantası problemi için ilk popülasyonun oluşturulması

Popülasyon	DNA dizileri	Sayısal karşılığı	Kutu için seçilecek elemanlar
P1	AATTCGGTTA	0011011110	3,4,6,7,8,9
P2	TTTGGCCACT	1111100001	1,2,3,4,5,10
P3	TAACAGGTAC	1000011100	1,6,7,8
P4	CCCTTACCCG	0001100001	4,5,10
P5	AAAACCCCTA	0000000010	9
P6	ATTAAAAAGC	0110000010	2,3,9
P7	GGCCCAAATG	1100000011	1,2,9,10

Adım 2: Popülasyondaki, seçimi belirlenen her bir elemanın kazanç ve ağırlık değerleri kullanılarak, uygunluk değerleri hesaplanmaktadır. Tablo 4.12'de oluşturulan popülasyon elemanlarının ağırlıkları, Denklem 4.8'de yerine konularak, kapasiteyi geçip geçmediği kontrol edilir. Denklem 4.9'da, kazanç değerleri yerine konularak, toplam kazanç değeri hesaplanır. Tablo 3.13'te, uygunluk değerleri gösterilmektedir.

Tablo 4.13. Örnek sırt çantası problemi için uygunluk değerlerinin bulunması

Popülasyon	DNA dizileri	Kutu için seçilecek elemanlar	Ağırlık toplamı	Kazanç Toplamı
P1	AATTCGGTTA	3,4,6,7,8,9	24+23+20+24+28+32=151	10+14+6+13+4+10=57
P2	TTTGGCCACT	1,2,3,4,5,10	26+17+24+23+28+25=143	10+7+10+14+17+15=73
P3	TAACAGGTAC	1,6,7,8	26+20+24+28=98	10+6+13+4=33
P4	CCCTTACCCG	4,5,10	23+28+25=76	14+17+15=46
P5	AAAACCCCTA	9	32	10
P6	ATTAAAAAGC	2,3,9	17+24+32=73	7+10+10=27
P7	GGCCCAAATG	1,2,9,10	26+17+32+25=100	10+7+10+15=42

Adım 3: Problem için belirlenen kapasite değerini aşan popülasyon elemanları, çözümden çıkarılmaktadır. Bu koşulu sağlamayan elemanlar çözümden çıkarıldığında, kalan elemanlar Tablo 4.14'de verilmektedir. Bu durumda, P1, P2, P3 ve P7 çözüm kümesinden çıkarılmaktadır.

Tablo 4.14. Örnek sırt çantası probleminde kapasite koşuluna uyan elemanlar

Popülasyon	DNA dizileri	Kutu için seçilecek elemanlar	Ağırlık toplamı	Kazanç Toplamı
P4	CCCTTACCCG	4,5,10	23+28+25=76	14+17+15=46
P5	AAAACCCCTA	9	32	10
P6	ATTAAAAAGC	2,3,9	17+24+32=73	7+10+10=27

Adım 4: Kalan popülasyon içerisinde, maksimum kazançlı elemanlar belirlenmektedir. Bir önceki adımda kalan elemanlar içerisinde, maksimum kazanç değeri 46 olan P4, problemin çözüm kümesi olmaktadır.

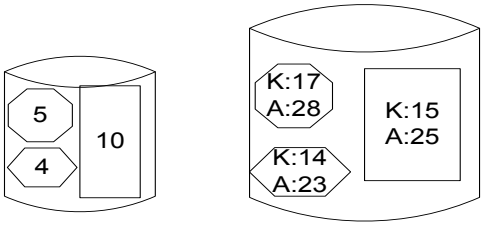
Adım 5: İlk popülasyona, çaprazlama uygulayarak, yeni popülasyon oluşturulup 2, 3 ve 4 adımları tekrarlanıp, maksimum kazanç değerleri belirlenmektedir.

Adım 6: İlk popülasyona, enzim ve virüs mutasyonu uygulanarak, yeni popülasyon oluşturulup 2, 3, 4 adımları tekrarlanıp, maksimum kazanç değerleri yeniden belirlenmektedir.

Adım 7: Maksimum işlem sayısına ulaşıldığında, yukarıda belirtilen adımlar sonucunda bulunan maksimum kazanç değeri, problemin çözümü olarak kabul edilmektedir.

Bilgisayar simülasyonu sonucunda, 4, 5 ve 10 numaralı elemanlar seçilerek çantaya konulmaktadır. Yapılan işlem sonucunda, maksimum kazanç değeri 46 olarak bulunmakta ve toplam ağırlık 76 olmaktadır. DNA hesaplama algoritması ile elde edilen çözüm sonucunda oluşan değerler, Şekil 4.8'de ayrıntılı olarak verilmektedir.

secilen_elemanlar : 4 5 10
maksimum_kazanc_degeri = 46
toplam_agirlik_degeri = 76



Şekil 4.8. Sırt çantası problemi için bulunan sonuçlar

4.4. Bölüm Değerlendirmesi

Literatürde yapılan uygulamalardan farklı olarak bu tez çalışmasında, DNA dizilerinin uzunlukları belirlenirken, problemi oluşturan değişken sayısı dikkate alınmakta, böylece hesaplama süresi kısaltılmaktadır. Örneğin, 2 baz uzunluğu ile kodlanacak bir problemin, gereksiz yere uzun diziler ile kodlanarak, kompleks bir yapı oluşturması engellenmiştir. İşlem hacmi sınırlı düzeyde tutularak, algoritmanın daha kısa sürede sonuçlanması sağlanmıştır. Uygulanan mutasyon yöntemi ile çözüm uzayı genişletilmiş ve daha doğru değerlerin bulunması amaçlanmıştır.

5. SİSTEM MODELLEMEDE DNA HESAPLAMANIN KULLANIMI

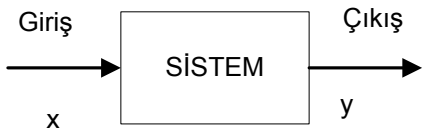
Gerçek hayatta anlaşılması güç olan sistemlerin farklı şekillerde tasarlanmasına modelleme denilmektedir. Modelleme yapılan sistem karmaşıktır ve o sistemin çalışması çoğunlukla net olarak anlaşılamamaktadır. Modelleme yapılırken, gerçek hayatta var olan büyük sistemler, daha küçük parçalar şeklinde tasarlanır ve insanların kolayca inceleme yapması sağlanır. Modelleme ile hedeflenen asıl amaç, karmaşık olan sistemlerin basitleştirilmesi ve parçalara ayırarak detaylı inceleme imkânı sağlanmasıdır. İstenilen bir sistem modellendiğinde farklı şekillerde incelenebilir. Gerçekleştirilmek istenilen modeller, ana sistemlerin tüm özelliklerini yansıtmayabilir [42]. Bu nedenle, sadece sistemin basitleştirilmiş bir kopyası olarak kabul edilmelidir. Günümüzde modelleme, birçok alanda kullanılmaktadır. Mühendislik alanında, üretimi yapılacak nesnelerin önceden bir modeli oluşturularak detaylı inceleme yapılır. Üretimden sonra olması istenen özellikler, model üzerinde çalışılarak karara bağlanır. İnşaat alanında yapılan modellemede, mimar ve mühendisler öncelikle konutların bir modellemesini yaparlar. Bu model üzerinden konutun yapımı devam eder ve yine bu modelle konut alacak kişiler, sahip olacakları mülkün özelliklerini, fiziki görünümünü öğrenirler. Araba üretim fabrikalarında önce modelleme yapılıp sonra üretime geçilir. Modellemenin kullanıldığı diğer bir alan ise, askeri alanlardır. Atış poligonları, uçak ve helikopter kullanımı için önce bir model oluşturulur. Eğitim yapacak kişiler, model üzerinde çalışarak gerçeğe yakın eğitim alırlar. Böylece, ekonomik açıdan büyük bir avantaj sağlanır. Günümüzde modelleme yapılırken, çoğunlukla bilgisayar simülasyonundan yararlanılmaktadır. Bilgisayar üzerinde yazılımı yapılan sistemler, gerçeğe yakın modelleme imkânı sağlarlar. Bilgisayar simülasyon programlarından biri matlab/simulink yazılımıdır. Bu programla belirli ölçülerde modelleme imkânı olmaktadır. Bilgisayar ortamında modelleme, iki veya üç boyutlu olarak gerçekleştirilir. Yapılan ışıklandırma ve renklendirme çalışmaları ile, üç boyutlu modelleme birçok modelin gerçeğe yakın bir şekilde tasarlanmasını ve görülmesini sağlar.

Modelleme yapılırken, sistemlere ait birçok bilginin gösterilmesi ve hesaplama için kullanılması gerekir. DNA hesaplama, sahip olduğu veri saklama kapasitesi ve hızlı işlem yapma yeteneği ile, modelleme alanında sıklıkla kullanılmaktadır. Özellikle, denetleyici parametrelerinin tasarlanması ve ayarlanması alanında oldukça fazla çalışma

yapılmaktadır. Yapılan çalışmalardan görüleceği üzere, DNA molekülleri modelleme için etkili bir araç olmaktadır.

5.1. Genel Bilgiler

Bir sistemin istenilen şekilde hareket etmesini sağlamak için yapılan çalışmalara kontrol, kontrol edilmesi istenen birimlere kontrol sistemi denir. Kontrol sistemleri 3 temel birimden oluşmaktadır [42]. Bunlar giriş parametreleri, kontrolü istenen sistem ve çıkış parametreleridir. Şekil 5.1'de örnek bir sistem gösterilmiştir.



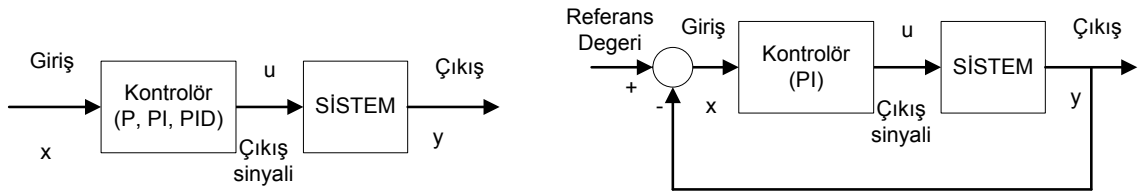
Şekil 5.1. Örnek bir sistem

Giriş kısmında, sistemi etkileyen giriş parametreleri belirlenmektedir. Giriş parametrelerini kontrol elemanının gönderdiği bir sinyal olarak kullanabiliriz. Sistem kısmında, kontrol edilmek istenen sistemin denklemi veya transfer fonksiyonu yazılır. Çıkış kısmında ise giriş parametresine göre sistemin verdiği cevap alınmaktadır. Sistemleri kararlı hale getirmek için kontrol elemanlarına ihtiyaç vardır. Kontrol elemanları, kontrol edilmek istenen sistemin istenilen ölçütlerde değerler üretmesi için kullanılmaktadır. Sistemleri kontrol etmek için 2 yöntem vardır. Bunlar açık-çevrim (open – loop) kontrol yöntemi ve kapalı-çevrim (closed – loop) kontrol yöntemidir.

Kontrol işareti sistemin çıkış değerlerinden etkilenmeyen sistemlere, açık çevrim sistemler denir. Bu tip sistemlerde; sonradan meydana gelen değişiklikler veya dışarıdan yapılan etkiler kontrol edilememektedir. Kontrol sistemleri için hata ve hatanın türevi giriş için kullanılmaktadır. Kontrol elemanının ürettiği sinyal sisteme verilerek sistemin verdiği yanıt bulunmaktadır.

Sistemlerin kontrolü ile ilgili olarak su deposu örnek verilebilir. Bu sistemde bir su pompası vardır. Bu pompa depoda su azaldığında sisteme su verir. Depo dolduğunda ise su verme işlemi durdurulur. Burada kontrol edilen eleman su pompasıdır.

Sistem çıkışının geri bildirim ile sistemin girişine bağlandığı sistemlere, kapalı çevrimli sistemler denir. Kapalı çevrim sistemlerde; sistemde sonradan oluşan etkiler anında kontrol edilerek düzeltilir ve sistemin kısa sürede kararlı hale gelmesi sağlanır. Kontrol sistemlerinin iyi bir performans sağlayabilmesi için kararlı durumun devam etmesi ve hatanın minimum düzeyde tutulması gerekir. Şekil 5.2'de örnek bir açık ve kapalı çevrim modeli verilmektedir.

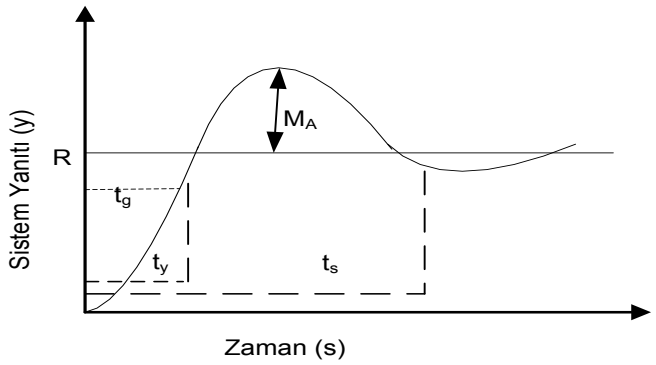


a) Açık Çevrimli Sistem

b) Kapalı Çevrimli Sistem

Şekil 5.2. Açık ve kapalı çevrimli sistemlerin gösterimi

Kapalı çevrim sistemler, açık çevrim sistemlere göre daha iyi bir performans sağlamakta ve daha kararlı bir yapı oluşturmaktadır. Sistemlerinin kararlı hale gelmesi için bazı değişkenler kullanılmaktadır. Bu değişkenler, Şekil 5.3'de grafik üzerinde gösterilmektedir [68, 73].



Şekil 5.3. Sistem yanıtı grafiği ve parametre gösterimi

Gecikme Zamanı (t_g): Sistemin ulaşacağı maksimum noktanın yarısına ulaşması için geçen süreye gecikme zamanı denilmektedir.

Yükselme Zamanı (t_y): Sistemin ulaşacağı maksimum noktanın %10'dan %90'ına, %5'ten %95'ine veya %0'dan %100'üne gelmesi için geçen süreye denir.
Maksimum Aşım (M_A): Sistemin ulaşacağı maksimum değer ile referans değeri arasındaki farka denir.
Oturma Zamanı (t_s): Sistem salınımının, sistem için belirlenen uygun değer aralıklarına ulaştığı zamana oturma zamanı denir. Oturma zamanının belirlenmesinde kullanılan değerler sistemlere göre farklılık göstermekle birlikte, genellikle %2 ile %3 olarak kabul edilir.

Kararlı bir sistemde gecikme zamanı, yükselme ve oturma zamanı ile maksimum aşım değerlerinin minimum düzeyde tutulması sağlanır. Sistemin hata değerini e ile referans değerini r ile ve sistemin çıkışını y ile gösterirsek hatanın hesaplanması için denklem 5.1 kullanılır. 4.1 denkleminde r değeri sistem için kabul edilen referans değeri olup sistemin bu değere yakın bir sonuç üretmesi hedeflenmelidir. Referans değerine yakın veya bu değeri üreten sistemler kararlı olarak kabul edilir. Denklem 5.1'de verilen y, çıkış değeri olup geri bildirim için sisteme gönderilmekte ve hata değerinin tespit edilmesi için kullanılmaktadır.

$$e(t) = r(t) - y(t) \tag{5.1}$$

Sistemin kararlı halde kalması için hataya bağlı olarak tanımlanan ve literatürde sıklıkla kullanılan fonksiyonların minimum düzeyde tutulması gerekir. Hataya bağlı bu fonksiyonlar aşağıda sırasıyla açıklanmaktadır.

ISE: Hatanın karesinin integralidir. Hatanın performansı hakkında bilgi verir. Genellikle büyük hataların sistemi kararsız hale getirdiği durumlarda kontrol için önemlidir. Denklem 5.2'de matematiksel gösterimi verilmektedir.

$$ISE = \int_0^T e(t)^2 dt \tag{5.2}$$

IAE: Hatanın mutlak değerinin integralidir. Genellikle küçük değerlerdeki hataların önemli olduğu sistemlerde kullanılır. Denklem 5.3'te matematiksel gösterimi verilmektedir.

$$IAE = \int_0^T |e(t)|dt \tag{5.3}$$

ITAE: Hatanın mutlak değeri ile zamanın çarpımının integralidir. Sistemin başlangıç noktasında meydana gelebilecek büyük hataların etkisinin azaltılması gereken durumlarda kullanılır. Denklem 5.4'te matematiksel gösterimi verilmektedir.

$$ITAE = \int_0^T t|e(t)|dt \tag{5.4}$$

ITSE: Hatanın karesi ile zamanın çarpımının integralidir. Denklem 5.5'te matematiksel gösterimi verilmektedir.

$$ITSE = \int_0^T te(t)^2 dt \tag{5.5}$$

5.2. PI Denetleyicilerin DNA Hesaplama ile Ayarlanması

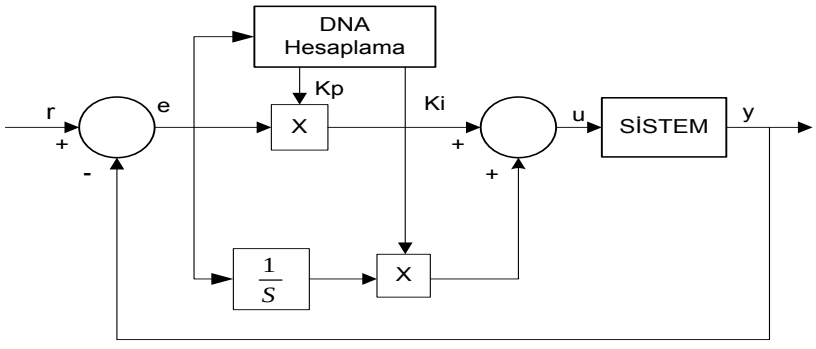
Uzun zamandan beri endüstri alanındaki sistemlerin kontrolünde, PI denetleyiciler kullanılmaktadır [45, 47, 48]. PI denetleyicilerin kullanılmasıyla; sistemlerin kontrolü, kararlılığı ve ekonomik alandaki verimliliğin artırılması amaçlanmaktadır. Bu amaç doğrultusunda sistemlerde PI kullanıldığında, amaçlanan hedeflere uygun sonuçlar üretildiği görülmüştür. PI denetleyiciler oransal sabit (P) ve integral sabiti (I) olarak kabul edilen iki temel yapıdan oluşmakta ve endüstri alanındaki sistemlerin kontrol edilmesi için kullanılmaktadır [46]. Bu sistemlerde kontrol edilecek bir parametre belirlenir ve bu parametrenin değerleri belirli aralıklarda tutulmaya çalışılır. Şekil 5.1'de PI tasarımında kullanılan bir sistemin genel yapısı gösterilmektedir. Şekilde verilen e hata değerini, R sistem için öngörülen set değerini ve y ise sistemin çıkış değerini göstermektedir. Denklem 5.6 PI denetleyicilerin ayarlanması için kullanılmaktadır.

$$u(t) = Kpe(t) + Ki\int e(t)d(t) \tag{5.6}$$

Denklem 5.6'da verilen Kp oransal sabit ve Ki integral sabitidir. Oransal sabit, kontrol sistemlerindeki yükselme zamanını azaltıcı etki yapmakta fakat hata değerini yok edememektedir. İntegral sabiti ise hata değerini yok etmektedir. Oransal ve integral sabiti birlikte kullanılarak sistem kararlı hale getirilmektedir. Eğer oransal ve integral sabitleri kullanıldığı halde sistem kararlı hale gelmezse türev sabiti olan Kd sisteme eklenir. Türev sabiti, oturma zamanını ve ani tepkileri azaltıcı etki göstermektedir. Zorunlu olmadıkça bu üç parametre birlikte kullanılmamalı. Çünkü üç parametrenin birlikte ayarlanması çok zor olmaktadır.

5.2.1. DNA Hesaplama Çözümü

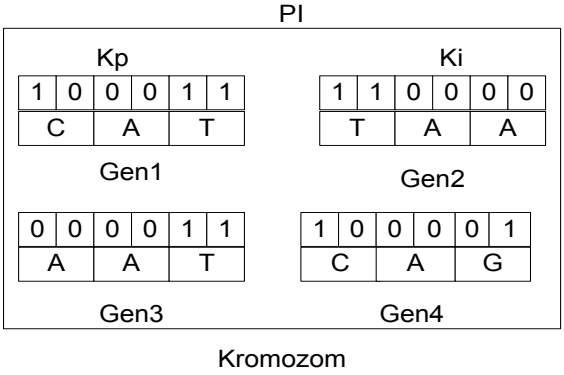
Sistemlerin kontrolü için denetleyici parametreleri kullanılmaktadır. Bu parametrelerin ayarlanması çoğunlukla zor olmaktadır. Bu nedenle denetleyici parametrelerinin ayarlanması için, optimizasyon algoritmaları geliştirilmiştir. Bu algoritmalar genellikle, PI parametrelerinin ayarlanmasında kullanılmıştır. PI, sistemin çıkışı ile referans değeri arasında oluşan hata değerini minimum düzeyde tutmak için tasarlanmaktadır. Yapılan birçok bilimsel uygulamada, PI parametreleri optimizasyon algoritmaları kullanılarak ayarlanmaktadır [45-47, 52]. Bu çalışmada, bir DC motorun pozisyon kontrolü için PI parametrelerinin DNA hesaplama ile optimizasyonu gerçekleştirilmiş olup, optimizasyon işlemi için Şekil 5.4'de verilen DNA-PI modeli kullanılmıştır.



Şekil 5.4. Önerilen DNA – PI modeli

Şekil 5.4'de görüldüğü üzere, sistem DNA hesaplama tabanlı olarak modellenmekte ve parametrelerin kodlanması için DNA dizileri kullanılmaktadır. Önerilen modelde Kp ve Ki

değişkenleri 3 baz uzunluğunda DNA dizileri ile kodlanmaktadır. Değişkenlerin alacağı değerler A, G, C ve T bazları kullanılarak, DNA dizileri ile rastlantısal olarak oluşturulmakta, sıra ile 0, 1, 2 ve 3 değerleri kullanılarak sayısal verilere çevrilmekte ve sisteme gönderilmektedir. Örneğin AAA=0, TTT=63 olarak alınmıştır. DNA hesaplama algoritması ile kodlama yapılırken Kp için 6 bit ve Ki için 6 bit olmak üzere, popülasyonun her bir elemanı 12 bit veri ile gösterilmiştir. Şekil 5.5'te örnek kodlama gösterilmektedir.



Şekil 5.5. PI parametrelerinin DNA hesaplama ile kodlanması

Optimizasyon uygulamalarında, PI elemanlarının değişim aralığı sonuçları önemli ölçüde etkilemektedir. Bu çalışmada Kp için 0-63 aralığı, Ki için 0-4 aralığı kullanılmaktadır. Aralık seçimi uygulamacının tecrübesi ve problem için deneme yanılma yöntemi ile belirlenir. DNA hesaplama ile yapılan kodlama örneği Tablo 5.1'de verilmektedir.

Tablo 5.1. PI parametrelerinin DNA hesaplama ile kodlanması

	DNA Kod Sistemi	4'lü Kod Sistemi	2'li Kod Sistemi	Onluk sisteme çeviri	Kp ve Ki Minimum Değeri	Kp ve Ki Maksimum Değeri
	A, G, C, T	0, 1, 2, 3	00, 01, 10, 11			
Kp	AGT, TAA, TTT	013, 300, 333	000111, 110000, 111111	0*16+1*4+3*1=7 3*16+0*4+0*1=48 3*16+3*4+3*1=63	0	63
Ki	GTT, CCA, AGC	133, 220, 012	011111, 101000, 000110	1*1+3*1/4+3*1/16=1.937 2*1+2*1/4*0*1/16=2.5 0*1+1*1/4+2*1/16=0.375	0	4

DNA hesaplama algoritmasının iki farklı uygulama şekli vardır. Çözelti ortamında yapılan uygulama maliyetli ve gerçekleştirilmesi zordur. Bu nedenle önerilen modelde, sayısal DNA hesaplama algoritması kullanılmaktadır. Sayısal DNA hesaplama algoritması

genetik algoritma ile benzerlik göstermektedir. Ancak DNA hesaplama algoritması, genetik algoritmadan farklı olarak iki yeni mutasyon işlemine sahiptir: Enzim ve virüs mutasyonu. Enzim mutasyonu, herhangi bir DNA dizisinden bir veya daha fazla DNA parçasının silinmesi demektir. Virüs mutasyonu ise herhangi bir DNA dizisine bir veya daha fazla yeni DNA parçası eklemektir. Enzim ve virüs mutasyonu, sistem performansı olumsuz duruma geldiğinde popülasyonun değişmesini sağlamaktadırlar. Popülasyonun sürekli değişmesi, yerel optimum noktalara takılmadan genel arama yapma olanağı sağlayıp, daha geniş bir çözüm alanı meydana getirir. Bu mutasyonların en büyük riski ise çözüm olabilecek bir elemanın silinmesidir.

Bu çalışmada, PI parametrelerinin ayarlanması için DNA hesaplama algoritmasını uygulayarak matlab 7.0 ile m-file yazılmıştır. Simülasyon sonuçlarının bulunması için Şekil 5.4'te verilen model simülink ortamında oluşturulmuştur. Simülasyon işleminde, DNA hesaplama için kullanılan parametre değerleri Tablo 5.2'de verilmektedir.

Tablo 5.2. Simülasyon için kullanılan DNA hesaplama parametreleri

Parametre	Değeri
Populasyon Büyüklüğü	60
Maksimum İşlem Sayısı	30
Çaprazlama oranı	%100
Enzim ve Virüs Oranı	0.3

Aşağıda m-file için kullanılan DNA hesaplama algoritmasının, algoritmik adımları ayrıntılı olarak açıklanmaktadır:

Adım 1: DNA dizileri ile rastlantısal olarak ilk popülasyon oluşturulur.

Adım 2: Oluşan diziler, yukarıda açıklandığı üzere önce 4'lü sisteme sonra onluk sisteme çevrilerek sayısal verilere dönüştürülür. Kodlama ve dönüştürme işlemleri, Tablo 5.1'de ayrıntılı olarak verilmektedir. Sayısal verilere dönüştürülen popülasyon elemanları, simülasyon ortamına gönderilerek sistemde oluşan hata değerleri belirlenir.

Adım 3: Simülasyondan gelen hata değeri, uygunluk fonksiyonunda yerine konularak uygunluk değeri elde edilir. Uygunluk değerini minimum yapan popülasyon elemanları, yeni Kp ve Ki değerleri olarak kullanılır. Uygunluk değerinin belirlenmesi için denklem 5.7 seçilmiştir. Bu uygunluk fonksiyonu yukarıda açıklandığı üzere uygulayıcının tercihine

göre değiştirilebilir. Literatürde PI parametrelerinin belirlenmesi için kullanılan birçok denklem vardır. Bunlardan bazıları yukarıda açıklanmıştır.

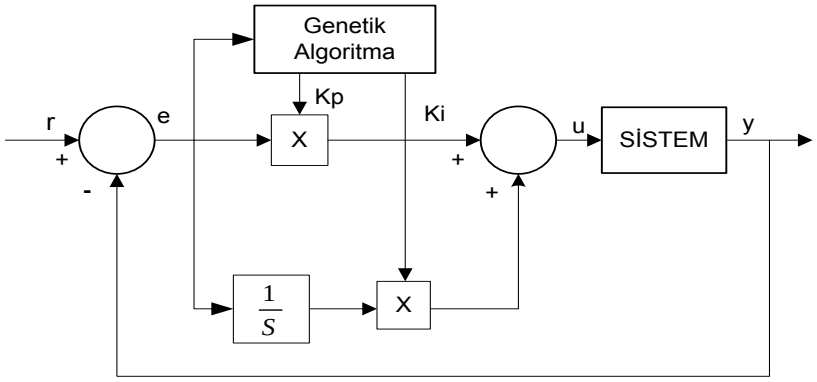
$$f(Kp,Ki)=sum(abs(e)) \quad (5.7)$$

Adım 4: İlk oluşturulan popülasyon elemanları çaprazlama işlemine tabi tutularak, yeni popülasyon oluşturulur. Adım 2 ve adım 3 kullanılarak yeni Kp ve Ki belirlenir.
Adım 5: Popülasyona enzim ve virüs mutasyonu uygulanarak yeni popülasyon oluşturulur. Adım 2 ve adım 3 uygulanarak yeni Kp ve Ki değerleri belirlenir.
Adım 6: Yukarıdaki adımlarda bulunan en iyi uygunluk değerleri karşılaştırılır. Hangi adımdaki uygunluk değeri minimum ise o adımda bulunan Kp ve Ki değerleri sisteme gönderilir.
Adım 7: Maksimum işlem sayısına ulaşılana kadar bu adımlar uygulanır. Maksimum işlem sonunda en uygun Kp ve Ki değerleri belirlenir.

5.2.2. Genetik Algoritma Çözümü

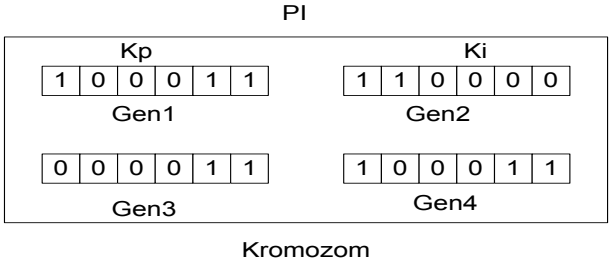
Son yıllarda, PI parametrelerinin optimizasyonu için birçok teknik uygulanmaktadır. Bu tekniklerden biride genetik algoritmadır. Genetik algoritma, doğal hesaplama özelliği ile hiçbir karmaşık matematiksel modele gerek duymaksızın optimum çözüme ulaşabilmekte olup, canlılardaki genetik bilginin özelliklerinden yararlanılarak geliştirilmiştir [73]. Bu özellikler, çaprazlama ve mutasyon olup, doğal yaşamın değişerek ve gelişerek devam etmesini sağlamaktadırlar. Popülasyon büyüklüğü, mutasyon, çaprazlama oranı ve maksimum işlem sayısı genetik algoritmalar için önemli parametreler olup, her problemde değerlerinin iyi belirlenmesi gerekmektedir. Popülasyon büyüklüğü, uygulayıcı tarafından problemin zorluk ve karmaşık yapısı dikkate alınarak belirlenir. Bu parametrenin büyüklüğünün artırılması, çözüm alanını genişletmekle beraber hesaplama süresinin uzamasına neden olur. Çaprazlama uygulanırken, mevcut popülasyonun elemanları kullanılarak yeni elemanların oluşturulması sağlanır. Çaprazlama oranı artırıldığında, bazı çözüm elemanlarının kaybolma riski artar. Mutasyon uygulamasının amacı, popülasyondaki çeşitliliği artırarak, algoritmayı yerel optimum noktalardan kurtarmaktır. Bu bölümde, PI parametrelerinin ayarlanması için GA-PI modeli geliştirilmiştir. Şekil 5.6'da bu model verilmektedir. Yapılan çalışmada, öncelikle genetik algoritma için

rastlantısal olarak ilk popülasyon oluşturularak, PI parametreleri için kullanılmaktadır. Seçilen popülasyonda Kp için 0-63 aralığı ve Ki için 0-4 aralığı öngörülmüştür. Bu aralık uygulayıcıların deneyimleri ile ve deneme yanılma yöntemi ile belirlenmektedir. Genetik algoritma ile kodlama yapılırken ikili sayı sistemi kullanılmakta ve popülasyon 0-1 dizilimi ile oluşturulmaktadır. Problemi oluşturan her değişken gen ile kodlanmakta, genler birleştirilerek çözüm elemanları olan kromozomlar elde edilmektedir.



Şekil 5.6. Önerilen GA-PI modeli

Yapılan çalışmada, Kp ve Ki değişkenlerinin gösterimi için 6 bit uzunluğunda ikili diziler kullanılmıştır. Dizilerin 6 bit uzunluğunda olmasının nedeni Kp değişkenin alabileceği maksimum değer olan 63 sayısının ikili sistemde en az 6 bit ile ifade edilebilmesidir. Bu diziler 10'luk sisteme çevrilerek optimizasyon işleminde kullanılmaktadır. Kodlama işlemi, Şekil 5.7'de ayrıntılı olarak gösterilmektedir. Tablo 5.3'te Genetik algoritma ile PI parametrelerinin kodlanması ve gerçek değerlere dönüştürülmesi verilmektedir.



Şekil 5.7. PI parametrelerinin GA ile kodlanması

Tablo 5.3. PI parametrelerinin GA ile kodlanması

	Genetik algoritma ile kodlama	Gerçek değerlere çevirme (10'luk sistem)	Kp ve Ki değişkenlerinin alacağı minimum değer	Kp ve Ki değişkenlerinin alacağı maksimum değer
Kp	000010 000111 111000	0*32+0*16+0*8+0*4+1*2+0*1=2 0*32+0*16+0*8+1*4+1*2+1*1=7 1*32+1*16+1*8+0*4+0*2+0*1=56	0	63
Ki	101000 010101 000110	0*1+0*1/2+0*1/4+0*1/8+1*1/16+0*1/32=1.25 0*1+0*1/2+0*1/4+0*1/8+1*1/16+0*1/32=0.656 0*1+0*1/2+0*1/4+0*1/8+1*1/16+0*1/32=0.187	0	4

PI parametrelerinin ayarlanması için, genetik algoritma uygulanarak matlab 7.0 ile m-file yazılmış, simülasyon sonuçlarının bulunması için, Şekil 5.5'te verilen model simülink ortamında oluşturulmuştur. Simülasyon işleminde, genetik algoritma için kullanılan parametre değerleri Tablo 5.4'te verilmektedir.

Tablo 5.4. Simülasyon için kullanılan genetik algoritma parametreleri

Parametre	Değeri
Populasyon Büyüklüğü	60
Maksimum İşlem Sayısı	30
Çaprazlama oranı	%100
Mutasyon oranı	0.3

Aşağıda m-file için kullanılan genetik algoritmanın algoritmik adımları, ayrıntılı olarak açıklanmaktadır:

Adım 1: İkili sayı dizileri ile rastlantısal olarak ilk popülasyon oluşturulur.

Adım 2: Oluşan diziler yukarıda açıklandığı üzere, önce 2'li sisteme sonra onluk sisteme çevrilerek sayısal verilere dönüştürülür. Kodlama ve dönüştürme işlemleri, Tablo 5.3'te ayrıntılı olarak verilmektedir. Sayısal verilere dönüştürülen popülasyon elemanları, simülasyon ortamına gönderilerek, sistemde oluşan hata değerleri belirlenir.

Adım 3: Simülasyondan gelen hata değerleri, uygunluk fonksiyonunda yerine konularak, uygunluk değerleri elde edilir. Uygunluk değerini minimum yapan popülasyon elemanları,

yeni Kp ve Ki değerleri olarak kullanılır. Uygunluk değerinin belirlenmesi için denklem 5.7 seçilmiştir.

Adım 4: İlk oluşturulan popülasyon elemanları, çaprazlama işlemine tabi tutularak, yeni popülasyon oluşturulur. Adım 2 ve adım 3 kullanılarak yeni Kp ve Ki belirlenir.

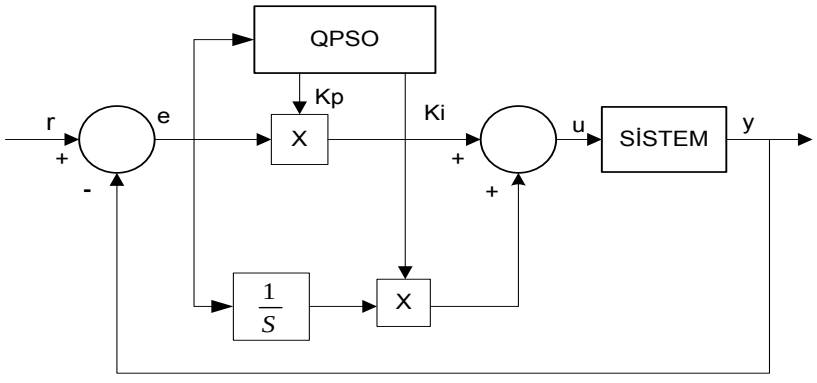
Adım 5: Popülasyona mutasyon uygulanarak, yeni popülasyon oluşturulur. Adım 2 ve adım 3 uygulanarak, yeni Kp ve Ki değerleri belirlenir.

Adım 6: Yukarıdaki adımlarda bulunan en iyi uygunluk değerleri karşılaştırılır. Hangi adımdaki uygunluk değeri minimum ise o adımda bulunan Kp ve Ki değerleri sisteme gönderilir.

Adım 7: Maksimum işlem sayısına ulaşılana kadar bu adımlar uygulanır. Maksimum işlem sonunda en uygun Kp ve Ki değerleri belirlenir.

5.2.3. Kuantum Parçacık Sürü Optimizasyonu Çözümü

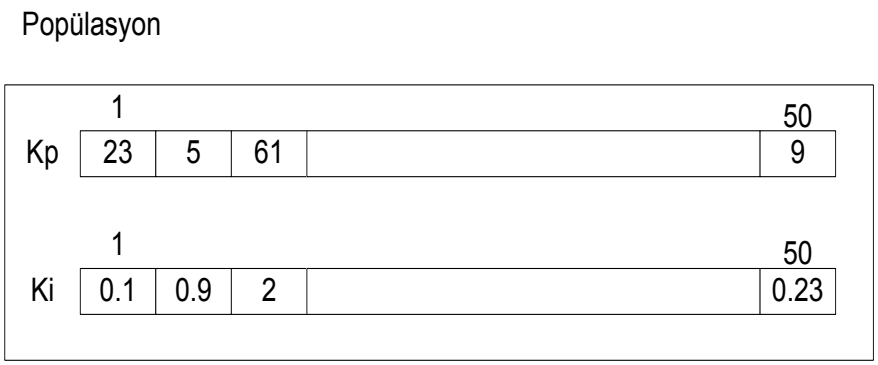
QPSO, doğadaki canlıların birbirleri ile olan sosyal durumu izlenerek geliştirilmiş bir optimizasyon algoritmasıdır [43]. Bu algoritma, literatür çalışmalarında bir çok problemin çözümü için kullanılmıştır. Popülasyon temelli algoritmalardan olan QPSO algoritmasında, bireyler birlikte hareket ederek sürüleri oluştururlar. Bu bölümde, PI parametrelerinin ayarlanması için QPSO-PI modeli önerilmiştir. Önerilen sistem için Şekil 5.8'de simülink modeli verilmektedir.



Şekil 5.8. Önerilen QPSO-PI modeli

Algoritma, başlangıçta rastgele oluşturulan bir popülasyon ile başlamakta, popülasyon elemanları Kp ve Ki değerlerini göstermektedir. Önerilen modelde, Kp için 0-63 aralığı, Ki

için ise 0-4 aralığı kullanılmakta ve gerçek değerler ile kodlama yapılmaktadır. Aralık seçimi, uygulayıcının deneyimi ile belirlenmektedir. Şekil 5.8'de gösterildiği üzere, Kp ve Ki değerleri QPSO algoritması ile bulunarak simülink ortamına gönderilmektedir. Simülink ortamında elde edilen hata değeri, algoritmaya gönderilerek uygunluk değerlerinin bulunması sağlanmaktadır. Şekil 5.9'da, Kp ve Ki parametrelerinin QPSO algoritması ile gösterimi verilmektedir. Burada görüldüğü üzere, popülasyon büyüklüğü kadar Kp ve Ki değerleri rastlantısal olarak oluşturulmaktadır.



Şekil 5.9. PI parametrelerinin QPSO algoritması ile gösterimi

PI parametrelerinin QPSO algoritması ile kodlanması, Tablo 5.5'te özet şeklinde verilmektedir. Burada, Kp ve Ki değerleri rastlantısal olarak ve gerçek değerler ile kodlanmaktadır.

Tablo 5.5. QPSO ile PI parametrelerinin kodlanması

	QPSO ile kodlama	Kp ve Ki için minimum değer	Kp ve Ki için maksimum değer
Kp	0, 12, 5, 7, 63, 55	0	63
Ki	0.01, 0.06, 0.2, 2, 1	0	4

Algoritma hazırlanırken, her parçacığın kendi elde ettiği en iyi bilgi (pbest) ile sürünün gerçekleştirdiği en iyi bilgi (gbest=min(pbest)) değişkenleri kullanılır. Bu iki değişken ile sürünün hareket noktası olan p değeri, Denklem 5.8 kullanılarak hesaplanmaktadır.

p= (Q1*pbest+Q2*gbest)/(Q1+Q2) (5.8)

Denklem 5.8'de verilen Q1 ve Q2 değişkenleri, 0-1 Aralığında üretilen rastgele sayıları ifade etmektedir. Sürünün bir sonraki konumunun belirlenmesinde, QPSO algoritması, gbest yerine sürünün en iyi aritmetik ortalaması olan mbest değişkenini kullanmaktadır. mbest değeri Denklem 5.9 kullanılarak hesaplanmaktadır.

$$mbest = \left(\sum_{m=1}^{n} pbest_i \right) / n \tag{5.9}$$

Denklem 5.9'da verilen n değeri, popülasyon büyüklüğünü göstermektedir. mbest değeri hesaplandıktan sonra, popülasyonun bir sonraki adımı x değeri, Denklem 5.10 kullanılarak hesaplanmaktadır.

$$x(i+1) = p \pm \beta . \left| mbest - x(i) \right| . \ln(\frac{1}{u}) \tag{5.10}$$

Denklem 5.10'daki β değişkeni, uygulayıcı tarafından belirlenecek bir değerdir. u değişkeni ise 0-1 aralığında değer alan bir başka değişkendir.

PI parametrelerinin ayarlanması için, QPSO algoritması uygulanarak, matlab 7.0 ile m-file yazılmaktadır. Simülasyon sonuçlarının bulunması için, Şekil 5.4'te verilen model simülink ortamında oluşturulmaktadır. Simülasyon işleminde, QPSO için kullanılan parametre değerleri Tablo 5.6'da verilmektedir.

Tablo 5.6. Simülasyon için kullanılan QPSO parametreleri

Parametre	Değeri
Populasyon Büyüklüğü	60
Maksimum İşlem Sayısı	30
c1, c2	0.1, 2
B	0.5

Aşağıda, m-file için kullanılan QPSO algoritmasının, algoritmik adımları ayrıntılı olarak açıklanmaktadır:

Adım 1: Kp ve Ki için ilk popülasyon rastlantısal olarak oluşturulur.

Adım 2: Popülasyon elemanları simülasyon ortamına gönderilerek, sistemde oluşan hata değeri belirlenir.

Adım 3. Simülasyondan gelen hata değeri uygunluk fonksiyonunda yerine konularak, uygunluk değerleri elde edilir. Uygunluk değerini minimum yapan popülasyon elemanları, yeni Kp ve Ki değerleri olarak kullanılır. Uygunluk değerinin belirlenmesi için, denklem 4.7 seçilmiştir.

Adım 4: Bulunan uygunluk değerleri, parçacıkların yerel konum değerlerini verir. Her bir parçacığa ait bu yerel değerler karşılaştırılır ve uygunluk değeri en iyi olan parçacığın konumu, popülasyonun yeni konumu olarak alınır.

Adım 5: Popülasyonun aritmetik ortalama değeri Mbest, Denklem 5.9 kullanılarak bulunur. Eğer bulunan değer önceki değerlerden iyi ise, yeni değer Mbest olarak değiştirilir.

Adım 5: Popülasyonun en iyi konumu, Denklem 5.10 ile hesaplanır.

Adım 6: Parçacıkların yeni pozisyonu, Denklem 5.8 kullanılarak güncellenir.

Adım 7: Maksimum işlem sayısına ulaşılana kadar 2, 3, 4, 5 ve 6 adımları tekrarlanır. Maksimum işlem sayısına ulaşıldığında, algoritma durdurulur. En iyi elemanlar seçilerek optimum çözüm elde edilir.

5.2.4. Simülasyon Sonuçları

Bu bölümde ve diğer bölümlerde, önerilen yöntemin performans ölçümü için Denklem 5.11 ile verilen bir DC motorun pozisyon kontrolünü modelleyen transfer fonksiyonu simülasyon sonuçları için kullanılmıştır.

$$G(s) = \frac{R(s)}{Ha(s)} = \frac{Yn}{s.\left[\ Ta + s.Ca\ .\ Dn + s.Gn\ + Yn.Yb\right]} \tag{5.11}$$

$$G(s) = \frac{2.2}{8.96e-6s^3 + 7.27e-3s^2 + 0.945s}$$

Yukarıda transfer fonksiyonu verilen DC motora ait parametreler, Tablo 5.7'de değerleri ile birlikte verilmektedir.

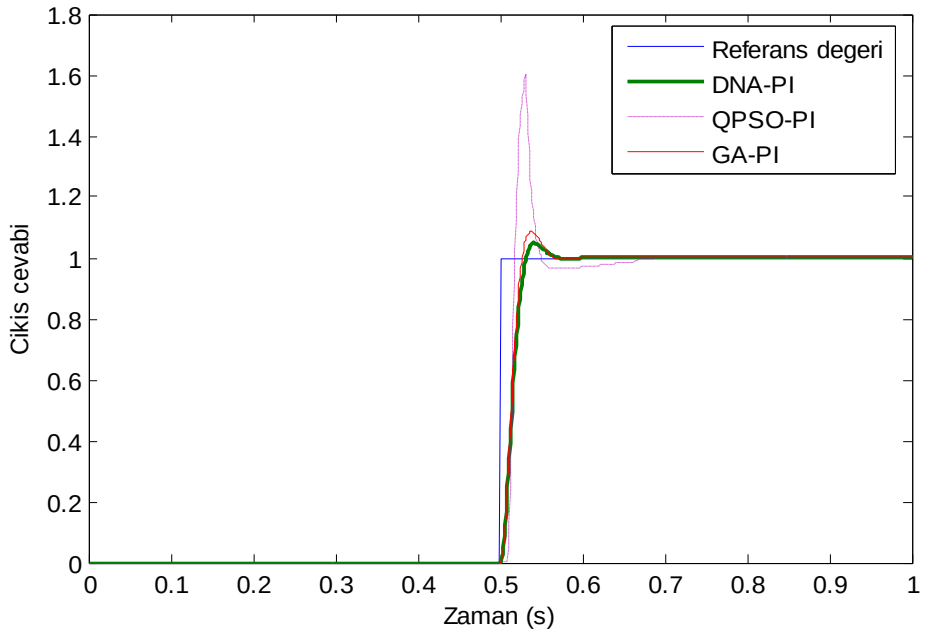
Tablo 5.7. DC motor parametre değerleri

Parametre	Aldığı değer
Ca	5.27 mH
Ta	3.9 Ω
Gn	0.0017 $Kgfcm^{-1}sec^{-2}$
Dn	0.121 $Kgfcm^{-1}sec^{-1}$
Yn	2.2 $Kgfcm^{-1}A^{-1}$
Yb	22.5 $Vkrpm^{-1}$

Yapılan uygulamada, PI parametrelerinin optimizasyonu için DNA hesaplama, genetik algoritma ve QPSO algoritması kullanılmıştır. PI parametrelerinin optimize edilmesinde, birçok uygunluk fonksiyonu kullanılmakla beraber, bu çalışmada hatanın mutlak değerinin toplamı, uygunluk fonksiyonu olarak seçilmiştir. Çok küçük değerlerde hatalara dahi duyarlı olan bu fonksiyonun kullanımı ile üst aşım, yükselme ve oturma zamanlarının daha iyi sonuç vermesi hedeflenmiştir.

Simülink çalışmasında kullanılan tüm algoritmalar için, popülasyon büyüklüğü 80, maksimum işlem sayısı 20 ve referans değeri 1 olarak alınmıştır. Kp ve Ki değerlerinin tespiti için Denklem 5.7 uygunluk fonksiyonu olarak kullanılmıştır. Optimizasyon algoritmaları, çevrimiçi ve çevrimdışı olarak sisteme uygulanmıştır. Çevrimiçi uygulamada, değerler online olarak sisteme gönderilip sonuçlar izlenmektedir. Çevrimdışı uygulamada ise, önce Kp ve Ki değerleri elde edilip, sisteme uygulanmıştır.

Çevrimiçi olarak yapılan uygulamada, algoritmalarda oluşturulan popülasyon elemanları sisteme gönderilerek PI elemanlarının belirlenmesi sağlanmıştır. Programın birçok kez çalıştırılması sonucu sistemin ürettiği sonuçlar, Tablo 5.7 ve Şekil 5.9'da verilmektedir. Tablo 5.7'de verildiği üzere, DNA hesaplama algoritması ile Kp 15, Ki 1.7188, yerleşme zamanı 0.01 sn ve maksimum aşım %12 olarak bulunmuştur. Diğer algoritmaların çalıştırılması ile bulunan sonuçlarda, karşılaştırma amaçlı olarak Tablo 5.7'de verilmektedir. Sayısal verilerden görüleceği üzere, DNA hesaplama algoritması kullanıldığında, diğer algoritmalara göre daha az aşım oluşmakta ve QPSO algoritmasına göre daha iyi yerleşme zamanı elde edilmektedir. Şekil 5.10, algoritmalar ile üretilen değerlerin çevrimiçi olarak simülink modeline gönderilmesi sonucu, sistemin oluşturduğu çıkış değerlerinin çizdirilmesi ile elde edilmiştir. Tablo 5.8 ile verilen yerleşme zamanı ve maksimum aşım, Şekil 5.10 üzerinde net olarak görülmektedir.



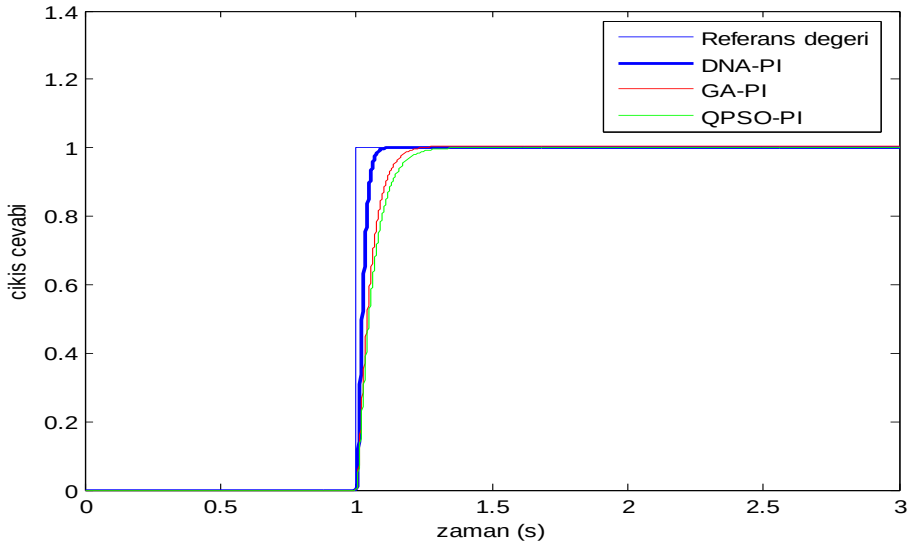
Şekil 5.10. Çevrimiçi sistem için simülasyon sonuçları

Tablo 5.8. Çevrimiçi sistem için simülasyon sonuçları

Algoritma	Kp değeri	Ki değeri	Yerleşme Zamanı	Maksimum Aşım %
DNA-PI	15	1.7188	0.08	14
GA-PI	12	1.8125	0.08	18
QPSO-PI	30	3.5234	0.1	60

Çevrimdışı olarak yapılan uygulamada, matlab programı ile m-file yazılıp, Kp ve Ki değerleri tespit edilmiştir. Kp ve Ki değerlerinin tespiti için, her bir iterasyonda, uygunluk fonksiyonun elde ettiği uygunluk değerleri kullanılmıştır. Uygunluk fonksiyonun, minimum değere ulaştığı iterasyon noktasında kullanılan Kp ve Ki değerleri, optimize edilmiş değerler olarak seçilmiştir. Programın birçok kez çalıştırılması sonucu bulunan Kp ve Ki değerleri, simülink modelinde yerine konularak, sonuçların oluşturulması sağlanmıştır. Bulunan sayısal değerler, Tablo 5.9 ve Şekil 5.11'de gösterilmektedir. Tablo 5.8'de verildiği üzere, DNA hesaplama algoritması ile Kp 19, Ki 0.0781, yerleşme zamanı 0.1 sn ve maksimum aşım yaklaşık %0 olarak elde edilmiştir. Diğer algoritmaların çalıştırılması sonucu bulunan değerlerde, karşılaştırma yapılması için Tablo 5.9'da ayrıntılı olarak verilmektedir. Bu değerlerden görüldüğü üzere, DNA hesaplama

algoritması kullanılarak bulunan maksimum aşım ve yerleşme zamanı değerlerinin, diğer algoritmalar ile bulunan değerlere göre çok daha iyi olduğu görülmektedir.



Şekil 5.11. Çevrimdışı sistem için simülasyon sonuçları

Tablo 5.9. Çevrimdışı sistem için simülasyon sonuçları

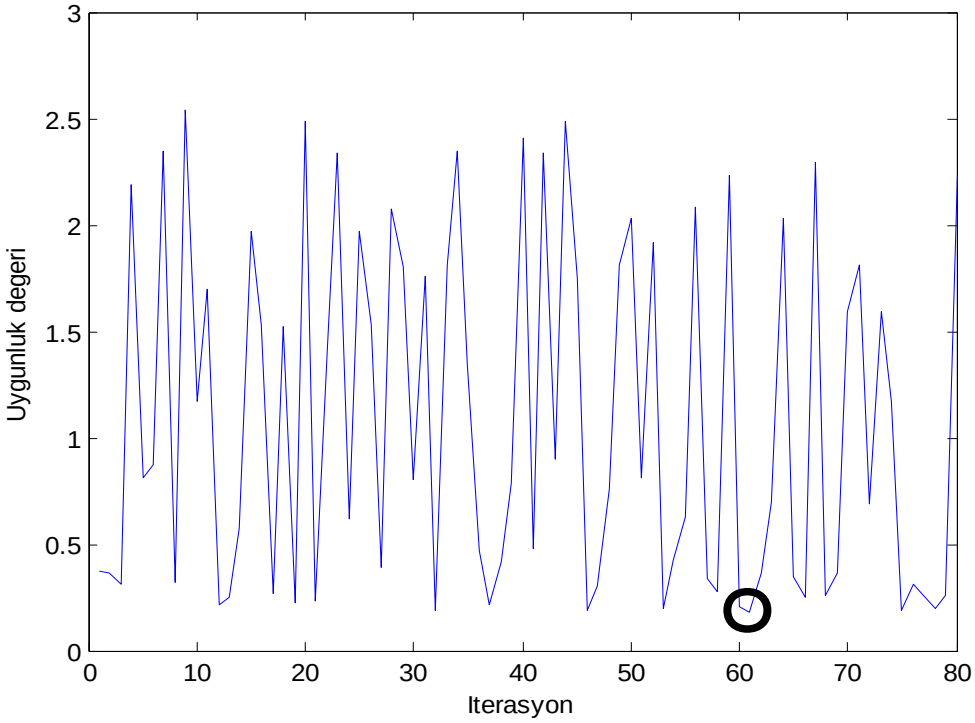
Algoritma	Kp değeri	Ki değeri	Yerleşme Zamanı	Maksimum Aşım %
DNA-PI	19	0.0781	0.1	%0
GA-PI	13	0.1250	0.3	%0
QPSO-PI	10.2803	0.7186	0.4	%0

DNA hesaplama algoritması ile kodlama yapılırken, Kp ve Ki değerleri 3 DNA bazı ile 6 bit veri kullanılarak kodlanmıştır. Popülasyonda, ilk olarak 80 birey kullanılmış, her birey 12 bit veri ile ifade edilmiştir. Bu 12 bit verinin, ilk 6 biti Kp için diğer 6 biti ise Ki için kullanılmıştır. Her bir iterasyonda, popülasyonun %30'u kadar enzim ve virüs mutasyonu uygulanarak, 80*0.3=24 bireyin değişimi sağlanmış ve popülasyon yenilenmiştir. Popülasyon elemanlarına ait uygunluk değerlerinin değişimi, Şekil 5.12 ve Tablo 5.10'da verilmektedir. Tablo 5.10 ayrıntılı incelendiğinde, en iyi uygunluk değerini 0.1799 değeri ile popülasyonun 61. elemanı sağladığı görülmektedir. Bunun için optimize edilmiş değerler Kp=19 ve Ki=0.0781 olarak seçilmiştir.

Tablo 5.10. DNA hesaplama ile elde edilen uygunluk, Kp ve Ki değerleri

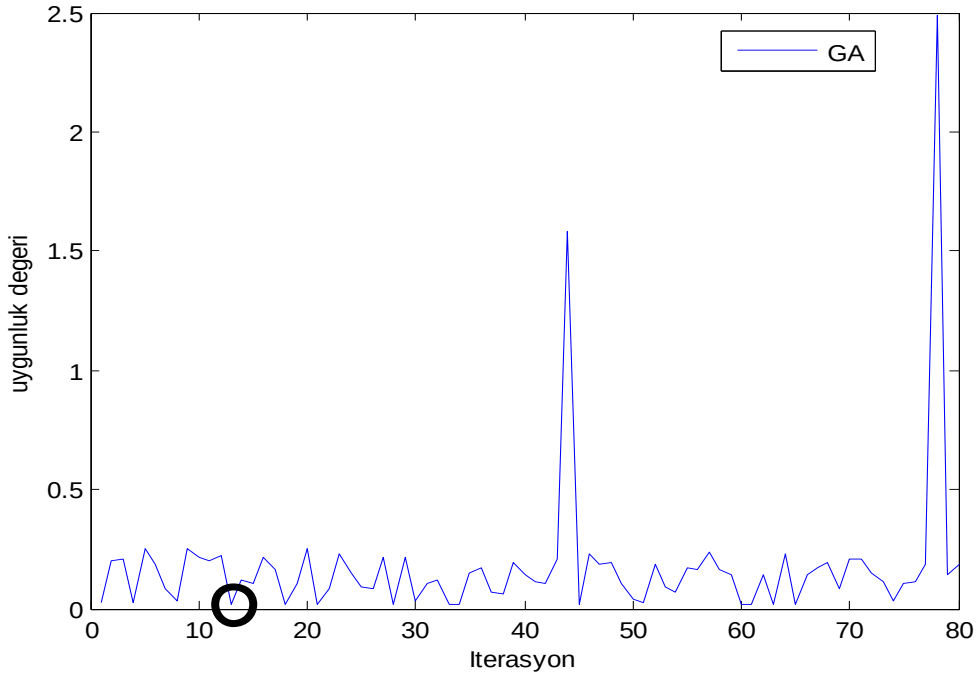
iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri	iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri
1	0.3753	8	1.0625	41	0.4790	26	3.9219
2	0.3623	24	1.4844	42	2.3375	59	0.1250
3	0.3134	23	1.0469	43	0.8998	5	3.1094
4	2.1869	56	2.3594	44	2.4861	62	0.7188
5	0.8105	32	1.7969	45	1.7510	48	1.6250
6	0.8713	33	1.9688	46	0.1919	18	1.7813
7	2.3444	59	3.9531	47	0.2990	9	0.4063
8	0.3211	23	2.0000	48	0.7576	31	3.2031
9	2.5384	63	1.2188	49	1.8130	49	3.0469
10	1.1714	38	1.7188	50	2.0298	53	3.3125
11	1.6977	47	1.6719	51	0.8114	32	2.0000
12	0.2193	12	0.0156	52	1.9147	51	0.3594
13	0.2541	13	1.9844	53	0.1997	18	2.3750
14	0.5705	28	0.9063	54	0.4283	25	3.9688
15	1.9722	52	0.7969	55	0.6230	29	0
16	1.5307	44	3.2656	56	2.0803	54	2.8125
17	0.2714	12	1.8281	57	0.3342	9	1.1094
18	1.5255	44	1.3906	58	0.2783	11	1.2656
19	0.2256	17	3.9219	59	2.2368	57	3.2969
20	2.4901	62	3.1406	60	0.2029	14	0.4375
21	0.2310	20	3.0156	**61**	**0.1799**	**19**	**0.0781**
22	1.4097	42	1.6250	62	0.3669	10	2.7969
23	2.3382	59	0.4844	63	0.6970	30	2.8750
24	0.6185	4	0.1719	64	2.0306	53	3.6875
25	1.9707	52	0.0938	65	0.3475	10	2.2969
26	1.5322	44	3.8125	66	0.2494	15	3.5625
27	0.3920	10	3.4531	67	2.2897	58	2.8125
28	2.0764	54	0.8906	68	0.2545	21	2.7969
29	1.8063	49	0.2031	69	0.3625	24	1.5156
30	0.8029	32	0.1094	70	1.5886	45	3.6875
31	1.7565	48	3.9063	71	1.8135	49	3.2500
32	0.1875	15	0.2188	72	0.6926	30	2.0000
33	1.8119	49	2.5781	73	1.5893	45	3.9531
34	2.3435	59	3.4063	74	1.1687	38	0.9531
35	1.3483	41	0.9375	75	0.1923	18	1.8125
36	0.4728	26	2.9531	76	0.3156	9	0.7344
37	0.2189	18	3.8438	77	0.2571	21	3.0625
38	0.4133	8	1.6875	78	0.1938	18	1.9219
39	0.7886	3	0.0781	79	0.2583	10	0.0938
40	2.4110	2	2.0938	80	2.2319	57	0.7656

Tablo 5.10 ve Şekil 5.12'de çözüm için seçilen popülasyon elemanları ve uygunluk değerleri verilmektedir. Uygunluk değerlerinin 0 ile 3 Aralığında değiştiği görülmektedir. Uygunluk fonksiyonunu minimum yapan 61. popülasyon elemanı çözüm olarak seçilmiştir. Fonksiyonun aldığı minimum değer 0.1799 olup bu değerin oluşmasını sağlayan Kp=19 ve Ki=0.0781 optimum değer olarak alınmıştır.



Şekil 5.12. DNA hesaplama algoritması uygunluk fonksiyonu değerleri

Genetik algoritma ile kodlama yapılırken, Kp ve Ki değerleri 6 bit veri kullanılarak kodlanmıştır. Popülasyonda ilk olarak 80 birey kullanılmış, her birey 12 bit veri ile ifade edilmiştir. Bu 12 bit verinin ilk 6 biti Kp için, diğer 6 biti ise Ki için kullanılmıştır. Her bir iterasyonda, popülasyonun %30'u kadar enzim ve virüs mutasyonu uygulanarak, 80*0.3=24 bireyin değişimi sağlanmış ve popülasyon yenilenmiştir. Genetik algoritma popülasyonuna ait uygunluk değerleri, Şekil 5.13 ve Tablo 5.11'de verilmiştir. Şekil 5.13 incelendiğinde, en iyi uygunluk değerini 0.0170 ile popülasyonun 13. elemanı sağlamıştır. 13. Elemanın kullanımı sonucu oluşan veriler, optimize edilmiş değerler olup, Kp=13 ve Ki=0.1250 olarak seçilmiştir. Diğer elemanların uygunluk değerleri incelendiğinde, daha büyük veriler elde edildiği, uygunluk değerlerinin, 0 ile 2.5 aralığında değiştiği görülmektedir.



Şekil 5.13. Genetik algoritma uygunluk fonksiyonu değerleri

Tablo 5.11. Genetik algoritma ile elde edilen uygunluk, Kp ve Ki değerleri

iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri	iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri
1	0.0274	22	1.1875	41	0.1168	38	0.8125
2	0.1974	52	1.5625	42	0.1049	36	0.9375
3	0.2077	54	1.2500	43	0.2075	54	0.3125
4	0.0286	10	0.7500	44	1.5837	0	0.0625
5	0.2485	62	0.3750	45	0.0171	18	0.1875
6	0.1860	50	0.1250	46	0.2285	58	0.2500
7	0.0859	4	1.2500	47	0.1862	50	1.0000
8	0.0320	8	0.1875	48	0.1972	52	0.6875
9	0.2486	62	0.8750	49	0.1048	36	0.6250
10	0.2183	56	0.3750	50	0.0375	8	1.0625
11	0.2035	2	1.3125	51	0.0228	12	0.3125
12	0.2195	2	1.6250	52	0.1863	50	1.2500
13	**0.0170**	**13**	**0.1250**	53	0.0910	4	1.5000
14	0.1171	38	1.5000	54	0.0692	30	1.8750
15	0.1053	36	1.9375	55	0.1747	48	0
16	0.2185	56	1.5625	56	0.1638	46	0.9375
17	0.1639	46	1.0625	57	0.2388	60	0.7500
18	0.0211	14	0.8125	58	0.1637	46	0.3125
19	0.1047	36	0.2500	59	0.1411	42	1.9375
20	0.2487	62	1.5000	60	0.0194	18	1.9375
21	0.0211	20	1.1250	61	0.0181	16	0.3750
22	0.0810	32	1.6250	62	0.1409	42	1.2500
23	0.2286	58	1.0625	63	0.0174	18	0.3750
24	0.1523	44	0.5625	64	0.2285	58	0.3750
25	0.0910	4	1.5000	65	0.0203	14	0.4375

Tablo 5.11'in devamı

iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri	iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri
26	0.0807	32	0.9375	66	0.1410	42	1.8750
27	0.2183	56	0.1875	67	0.1748	48	0.1875
28	0.0184	18	1.1875	68	0.1973	52	1.3125
29	0.2185	56	1.6250	69	0.0805	32	0.6875
30	0.0363	24	1.5625	70	0.2076	54	0.5625
31	0.1050	36	1.2500	71	0.2101	2	1.4375
32	0.1171	38	1.6875	72	0.1525	44	1.3125
33	0.0198	16	1.4375	73	0.1166	38	0.1250
34	0.0190	18	1.6250	74	0.0333	10	1.9375
35	0.1525	44	1.1875	75	0.1052	36	1.5625
36	0.1751	48	1.8125	76	0.1166	38	0.2500
37	0.0688	30	1.0625	77	0.1863	50	1.5000
38	0.0596	6	1.8750	78	2.4868	0	1.1875
39	0.1971	52	0.3125	79	0.1410	42	1.6875
40	0.1409	42	1.5000	80	0.1863	50	1.3125

QPSO ile hesaplama yapılırken, popülasyonda Kp için 40 ve Ki için 40 olmak üzere toplam 80 birey kullanılmıştır. Bu algoritma ile elde edilen, popülasyona ait uygunluk değerleri Şekil 5.14'te verilmiştir. Burada en iyi uygunluk, 78. elemana ait 0.1210 değeri olduğundan, optimize edilmiş Kp=10.2803 ve Ki=0.7186 değerleri olarak kabul edilmiştir.

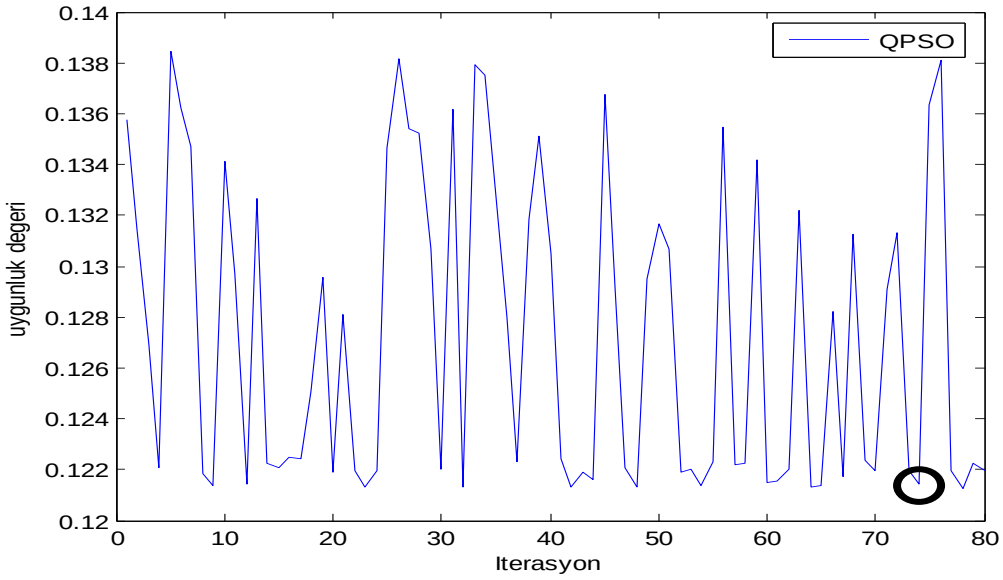
Tablo 5.12. QPSO ile elde edilen uygunluk, Kp ve Ki değerleri

iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri	iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri
1	0.1358	20.4093	1.7959	41	0.1224	21.2884	0.2485
2	0.1315	20.4093	1.1522	42	0.1213	17.2884	0.0783
3	0.1270	20.4093	0.4718	43	0.1219	21.2884	0.1705
4	0.1221	21.2884	0.1986	44	0.1216	21.2884	0.1204
5	0.1384	20.1597	2.0309	45	0.1368	20.1597	1.7816
6	0.1362	20.4093	1.8637	46	0.1292	20.4093	0.8078
7	0.1347	20.4093	1.6326	47	0.1221	21.2884	0.1947
8	0.1218	21.2884	0.1574	48	0.1213	19.2884	0.0748
9	0.1214	21.2884	0.0865	49	0.1295	20.4093	0.8547
10	0.1341	20.4093	1.5458	50	0.1316	20.4093	1.1723
11	0.1297	20.4093	0.8742	51	0.1307	20.4093	1.0291
12	0.1214	21.2884	0.0914	52	0.1219	21.2884	0.1722
13	0.1327	20.4093	1.3258	53	0.1220	21.2884	0.1826
14	0.1222	21.2884	0.2221	54	0.1214	21.2884	0.0862
15	0.1221	21.2884	0.1935	55	0.1223	21.2884	0.2365
16	0.1225	21.2884	0.2558	56	0.1355	20.4093	1.7469
17	0.1224	21.2884	0.2486	57	0.1222	21.2884	0.2160
18	0.1250	20.4093	0.1870	58	0.1222	21.2884	0.2205
19	0.1296	20.4093	0.8629	59	0.1342	20.4093	1.5573
20	0.1219	21.2884	0.1695	60	0.1215	21.2884	0.1030
21	0.1281	20.4093	0.6419	61	0.1216	21.2884	0.1168
22	0.1220	21.2884	0.1789	62	0.1220	21.2884	0.1844

Tablo 5.12'in devamı

23	0.1213	18.2884	0.0740	63	0.1322	20.4093	1.2587
24	0.1219	21.2884	0.1738	64	0.1213	17.2884	0.0759
25	0.1347	20.4093	1.6265	65	0.1213	18.2884	0.0816
26	0.1382	20.1597	1.9901	66	0.1282	20.4093	0.6633
27	0.1354	20.4093	1.7378	67	0.1217	21.2884	0.1428
28	0.1352	20.4093	1.7142	68	0.1313	20.4093	1.1179
29	0.1306	20.4093	1.0234	69	0.1223	21.2884	0.2385
30	0.1220	21.2884	0.1859	70	0.1220	21.2884	0.1787
31	0.1362	20.4093	1.8556	71	0.1291	20.4093	0.7926
32	0.1213	17.2884	0.0806	72	0.1313	20.4093	1.1272
33	0.1379	20.1597	1.9565	73	0.1220	21.2884	0.1788
34	0.1375	20.1597	1.8975	74	0.1214	21.2884	0.0954
35	0.1332	20.4093	1.4048	75	0.1364	20.1597	1.7247
36	0.1280	20.4093	0.6235	76	0.1381	20.1597	1.9778
37	0.1223	21.2884	0.2317	77	0.1220	21.2884	0.1785
38	0.1319	20.4093	1.2062	**78**	**0.1210**	**10.2803**	**0.7186**
39	0.1351	20.4093	1.6978	79	0.1222	21.2884	0.2200
40	0.1305	20.4093	1.0017	80	0.1220	21.2884	0.1813

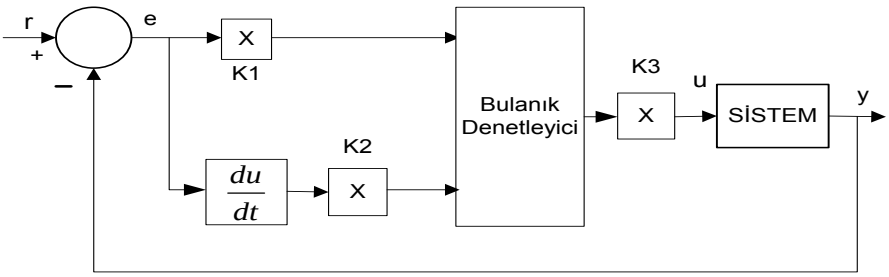
Tablo 5.12 ve Şekil 5.14'de, çözüm için seçilen popülasyon elemanları ve uygunluk değerleri verilmektedir. Uygunluk değerlerinin, 0 ile 0.14 aralığında değiştiği görülmektedir. Uygunluk fonksiyonunu minimum yapan, 78. popülasyon elemanı, çözüm olarak seçilmiştir. Fonksiyonun aldığı minimum değer 0.1210 olup, bu değerin oluşmasını sağlayan Kp=10.2803 ve Ki=0.7186 optimum değer olarak alınmıştır.



Şekil 5.14. QPSO algoritması uygunluk fonksiyonu değerleri

5.3. Bulanık Denetleyicilerin DNA Hesaplama ile Ayarlanması

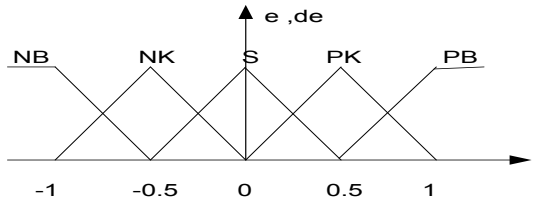
Basit ve sınırlı sayıda koşul içeren problemlerin çözümünde, klasik mantık kullanılarak kolayca sonuca ulaşılmaktadır. Ancak, çok karmaşık bir yapıya sahip, koşulları iç içe olan ve kişisel yorum gerektiren durumlarda klasik mantık yetersiz kalmakta ve çözüm zorlaşmaktadır [52, 74]. Klasik mantık ile çözümü zor olan problemler için, bulanık mantık tabanlı denetleyiciler geliştirilmiştir. Bulanık denetleyiciler, doğrusal olmayan, karmaşık ve çok parametreli sistemlerin optimizasyonunda oldukça yararlı olmaktadır. Bu denetleyiciler, araştırma ve uygulama alanında sıklıkla kullanılmakta ve ilgiyle takip edilmektedir. Bulanık denetleyicilerin, bulanık değerlerden oluşan kümeleri kullanması ile, kesin değerlerden oluşan çözüm kümelerine göre, daha geniş bir arama ve çözüm alanı sağlanmaktadır. Günlük hayatta kullanılan; çok küçük, küçük, orta, orta üstü, büyük, çok büyük gibi ifadeler bulanık değerler olarak ifade edilir. Bulanık mantık, hiçbir matematiksel denklem veya modele gereksinim duymaz. Problemi oluşturan, giriş ve çıkış değişkenlerini göstermek için üyelik fonksiyonlarını kullanır. Giriş değerlerine göre, çıkışın elde edilmesi için kural tabanı kullanılır. Kural tabanı, uzman kişilerin görüş ve önerileri ile tecrübelerinden yararlanılarak oluşturulur. Bulanık mantık, kesin değer ifade eden problemleri, bulanık değerlere dönüştürüp, gerçek hayattaki çözüm kümesine yakın sonuçlar üretir. Şekil 5.15'te, örnek bir bulanık denetleyicili sistem verilmektedir.



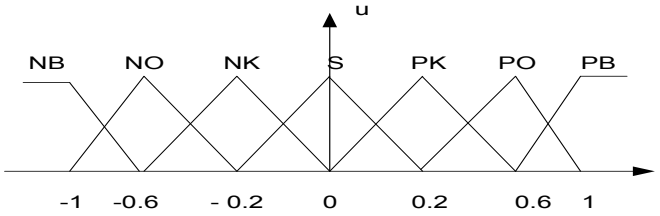
Şekil 5.15. Bulanık denetleyicili sistem

Bulanık denetleyici için, Şekil 5.16 ve Şekil 5.17'de, normalize edilmiş giriş ve çıkış üyelik fonksiyonları verilmektedir. Normalizasyon değerleri, DNA hesaplama ile ayarlanarak, sistem aktif hale getirilmektedir. Giriş ve çıkış üyelik fonksiyonları için, -1 ile +1 aralığında değişen değerler kullanılmaktadır. Şekil 5.16 ve Şekil 5.17 üzerinde verilen,

NB-Negatif Büyük, NO-Negatif Orta, NK-Negatif Küçük, S-Sıfır, PK-Pozitif Küçük, PO-Pozitif Orta ve PB-Pozitif Büyük bulanık değerleri ifade etmektedir.



Şekil 5.16. Hata (e) ve hatanın türevi (de) üyelik fonksiyonları



Şekil 5.17. Çıkış üyelik fonksiyonu (u)

Tablo 5.13. Bulanık denetleyici kural tablosu

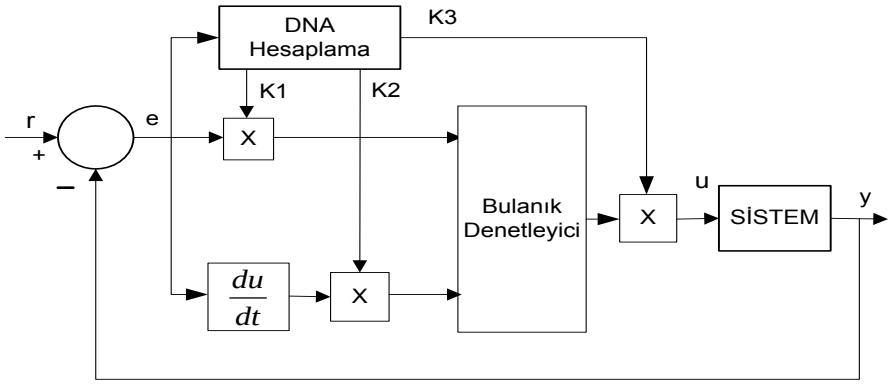
e / de	NB	NK	S	PK	PB
NB	NB	NB	NO	NK	S
NK	NB	NO	NK	S	PK
S	NO	NK	S	PK	PO
PK	NK	S	PK	PO	PB
PB	S	PK	PO	PB	PB

Tablo 5.13'te, giriş üyelik fonksiyonlarının kullanımı sonucu oluşacak çıkış değerleri verilmektedir.

5.3.1 DNA Hesaplama Çözümü

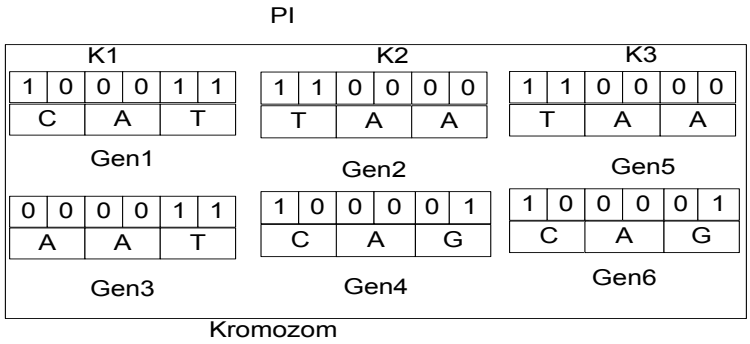
Bulanık denetleyiciler, PI denetleyiciler gibi, sistemlerin kontrol edilmesinde kullanılır. Bu denetleyicilerin, ölçekleme faktörlerinin ayarlanması ve kural tabanının oluşturulması, bir optimizasyon problemi olup, yapılan çalışmalarda bir çok algoritma geliştirilmiştir.

DNA hesaplama, bir optimizasyon algoritması olup bulanık denetleyici parametrelerinin ayarlanmasında kullanılabilmektedir. Bu algoritmada kullanılan DNA molekülleri, büyük oranda veri saklama ve paralel işlem yapma yeteneği ile, optimizasyon problemlerinin çözümünü kolaylaştırmaktadır. Bu çalışmada, Bulanık denetleyicilerin ölçekleme faktörlerinin ayarlanması için, DNA hesaplama algoritması geliştirilmiştir. Yapılan çalışma için, Şekil 5.18'de verilen DNA-Bulanık denetleyici modeli önerilmiştir.



Şekil 5.18. Önerilen DNA-bulanık denetleyici modeli

Şekil 5.18'de verilen, K1 ve K2 giriş, K3 ise çıkış ölçekleme faktörleri olup, DNA hesaplama ile ayarlanarak, bulanık denetleyici uyarlamalı hale getirilmektedir. DNA hesaplama ile, ölçekleme faktörü değerleri bulunarak sisteme gönderilmekte ve hata değerlerinin bulunması sağlanmaktadır. Bulunan hata değerleri, uygunluk fonksiyonunda yerine konularak uygunluk değerleri elde edilmekte ve uygunluk değeri en küçük olan, hata değerini oluşturan veriler, ölçekleme faktörlerinin yeni değerleri olarak kullanılmaktadır. Şekil 5.18'de görüldüğü üzere, sistem DNA hesaplama tabanlı olarak modellenmekte ve ölçekleme faktörlerinin kodlanması için DNA dizileri kullanılmaktadır. Önerilen modelde, K1, K2 ve K3 değişkenleri 3 baz uzunluğunda DNA dizileri ile kodlanmaktadır. Değişkenlerin alacağı değerler; A, G, C ve T bazları kullanılarak rastlantısal olarak oluşturulmakta ve sıra ile 0, 1, 2 ve 3 değerleri kullanılarak sayısal verilere çevrilip, sisteme gönderilmektedir. Şekil 5.19'da, bulanık denetleyicili sisteme ait ölçekleme faktörlerinin, DNA hesaplama ile kodlanması gösterilmektedir.



Şekil 5.19. Bulanık denetleyicili sistemin ölçekleme faktörlerinin DNA ile kodlanması

Optimizasyon problemlerinde, bulanık denetleyici parametrelerinin değişim aralığı sonuçları önemli ölçüde etkilemektedir. Bu çalışmada; K1, K2 ve K3 için 0-4 aralığı kullanılmıştır. Aralık seçimi, uygulamacının tecrübesi ve problem için deneme yanılma yöntemi ile belirlenir. DNA hesaplama algoritmasının, iki farklı uygulama şekli vardır. Çözelti ortamında yapılan uygulama, maliyetli ve gerçekleştirilmesi zordur. Bu nedenle, önerilen modelde sayısal DNA hesaplama algoritması kullanılmaktadır. Bulanık denetleyici parametrelerinin ayarlanması için DNA hesaplama algoritmasını uygulayarak, matlab 7.0 ile m-file yazılmıştır. DNA hesaplama ile yapılan kodlama örneği, Tablo 5.14'de verilmektedir.

Tablo 5.14. Bulanık denetleyiciler için DNA kodlama tablosu

	DNA Kod Sistemi	4'lü Kod Sistemi	2'li Kod Sistemi	Onluk sisteme çeviri	K1,K2 ve K3 için Minimum Değer	K1, K2 ve K3 için Maksimum Değer
	A, G, C, T	0, 1, 2, 3	00, 01, 10, 11			
K1	AGT, TAA, TTT	013, 300, 333	000111, 110000, 111111	0*1+1*1/4+3*1/16=0.4375 3*1+0*1/4+0*1/16=3 3*1+3*1/4+3*1/16=3.9375	0	4
K2	GTT, CCA, AGC	133, 220, 012	011111, 101000, 000110	1*1+3*1/4+3*1/16=1.9375 2*1+2*1/4*0*1/16=2.5 0*1+1*1/4+2*1/16=0.375	0	4
K3	CTT TAA ACG	133 100 021	101111 110000 001001	1*1+3*1/4+3*1/16=1.9375	0	4

Simülasyon işleminde DNA hesaplama için kullanılan parametre değerleri Tablo 5.15'de verilmektedir.

Tablo 5.15. Simülasyon için kullanılan DNA hesaplama parametreleri

Parametre	Değeri
Populasyon Büyüklüğü	60
Maksimum İşlem Sayısı	30
Çaprazlama oranı	%100
Enzim ve Virüs Oranı	0.3

Aşağıda m-file için kullanılan DNA hesaplama algoritmasının algoritmik adımları ayrıntılı olarak açıklanmaktadır:

Adım 1: DNA dizileri ile rastlantısal olarak ilk popülasyon oluşturulur.

Adım 2: Oluşan diziler, yukarıda açıklandığı üzere, önce 4'lü sisteme sonra onluk sisteme çevrilerek, sayısal verilere dönüştürülür. Kodlama ve dönüştürme işlemleri, Tablo 5.10'da ayrıntılı olarak verilmektedir. Sayısal verilere dönüştürülen popülasyon elemanları, simülasyon ortamına gönderilerek, sistemde oluşan hata değerleri belirlenir.

Adım 3: Simülasyondan gelen hata değerleri, uygunluk fonksiyonunda yerine konularak, uygunluk değerleri elde edilir. Uygunluk değerini minimum yapan popülasyon elemanları, yeni K1, K2 ve K3 değerleri olarak kullanılır. Uygunluk değerinin belirlenmesi için, denklem 5.7 seçilmiştir.

Adım 4: İlk oluşturulan popülasyon elemanları, çaprazlama işlemine tabi tutularak, yeni popülasyon oluşturulur. Adım 2 ve adım 3 kullanılarak, yeni K1, K2 ve K3 belirlenir.

Adım 5: Popülasyona, enzim ve virüs mutasyonu uygulanarak, yeni popülasyon oluşturulur. Adım 2 ve adım 3 uygulanarak, yeni K1, K2 ve K3 değerleri belirlenir.

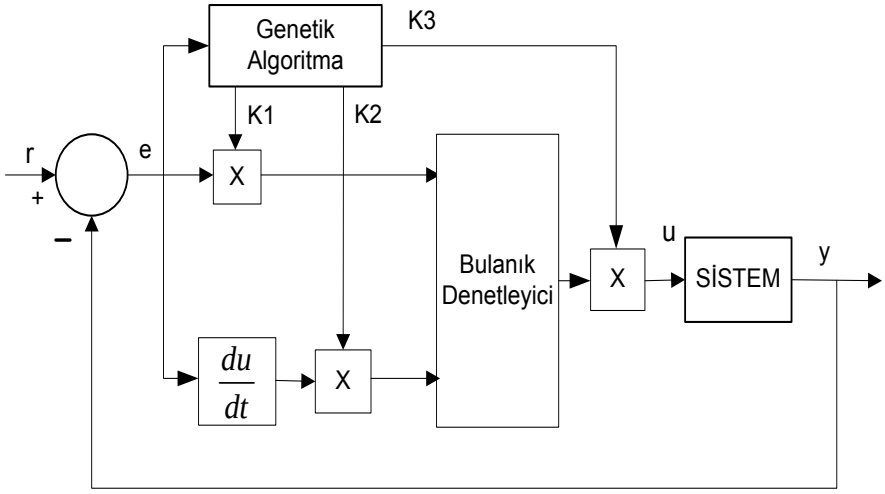
Adım 6: Yukarıdaki adımlarda bulunan en iyi uygunluk değerleri karşılaştırılır. Hangi adımdaki uygunluk değeri minimum ise, o adımda bulunan K1, K2 ve K3 değerleri sisteme gönderilir.

Adım 7: Maksimum işlem sayısına ulaşılana kadar, bu adımlar uygulanır. Maksimum işlem sonunda, en uygun K1, K2 ve K3 değerleri belirlenir.

5.3.2.Genetik Algoritma Çözümü

Genetik algoritmalar, doğal hesaplama özelliği ile hiçbir karmaşık matematiksel modele gerek duymaksızın, optimum çözüme ulaşabilmektedirler. Popülasyon büyüklüğü,

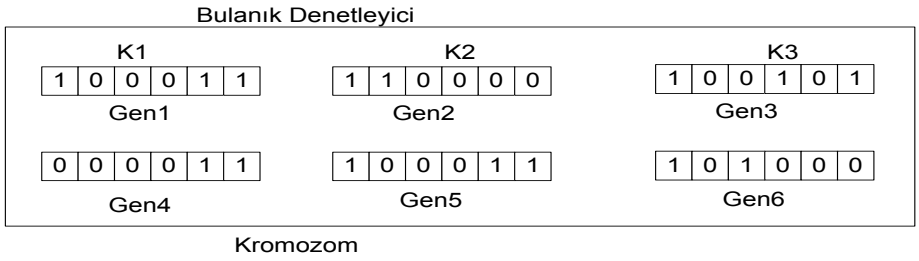
mutasyon, çaprazlama oranı ve maksimum işlem sayısı genetik algoritmalar için önemli parametreler olup, her problemde değerlerinin iyi belirlenmesi gerekmektedir. Bu bölümde denetleyici parametrelerinin ölçekleme faktörlerinin ayarlanması için, GA-bulanık denetleyici modeli önerilmiştir. Şekil 5.20'de, önerilen sistem için simülink modeli verilmektedir.



Şekil 5.20. Önerilen GA-bulanık denetleyici modeli

Bu modelde, öncelikle genetik algoritma için rastlantısal olarak ilk popülasyon oluşturularak, bulanık denetleyici ölçekleme faktörleri için kullanılmaktadır. Seçilen popülasyonda, K1, K2 ve K3 için 0-4 aralığı öngörülmüştür. Bu aralık, uygulayıcıların deneyimleri ve deneme yanılma yöntemi ile belirlenmektedir.

Genetik algoritma ile kodlama yapılırken, ikili sayı sistemi kullanılmakta ve popülasyon 0-1 dizilimi ile oluşturulmaktadır. Problemi oluşturan her değişken gen ile kodlanmakta genler birleştirilerek çözüm elemanları olan kromozomlar elde edilmektedir. Yapılan çalışmada, K1, K2 ve K3 değişkenlerinin gösterimi için 6 bit uzunluğunda ikili diziler kullanılmıştır. Bu diziler, 10'luk sisteme çevrilerek optimizasyon işleminde kullanılmaktadır. Kodlama işlemi Şekil 5.21'de ayrıntılı olarak gösterilmektedir.



Şekil 5.21. Bulanık denetleyicili sistemin ölçekleme faktörlerinin GA ile kodlanması

Tablo 5.16'da Genetik algoritma ile bulanık denetleyici ölçekleme faktörlerinin kodlanması ve gerçek değerlere dönüştürülmesi verilmektedir.

Tablo 5.16. Bulanık denetleyici parametrelerinin genetik algoritma ile kodlanması

	Genetik algoritma ile kodlama	Gerçek değerlere çevirme (10'luk sistem)	K1, K2 ve K3 değişkenlerinin alacağı minimum değer	K1, K2 ve K3 değişkenlerinin alacağı maksimum değer
K1	000010 000111 111000	0*32+0*16+0*8+0*4+1*2+0*1=2 0*32+0*16+0*8+1*4+1*2+1*1=7 1*32+1*16+1*8+0*4+0*2+0*1=56	0	63
K2	101000 010101 000110	0*1+0*1/2+0*1/4+0*1/8+1*1/16+0*1/32=1.25 0*1+0*1/2+0*1/4+0*1/8+1*1/16+0*1/32=0.656 0*1+0*1/2+0*1/4+0*1/8+1*1/16+0*1/32=0.187	0	4
K3	101000 010101 000110	0*1+0*1/2+0*1/4+0*1/8+1*1/16+0*1/32=1.25 0*1+0*1/2+0*1/4+0*1/8+1*1/16+0*1/32=0.656 0*1+0*1/2+0*1/4+0*1/8+1*1/16+0*1/32=0.187	0	4

Bulanık denetleyici parametrelerinin ayarlanması için genetik algoritma uygulanarak matlab 7.0 ile m-file yazılmıştır. Simülasyon sonuçlarının bulunması için, Şekil 5.14'te verilen model, simülink ortamında oluşturulmuştur. Simülasyon işleminde, genetik algoritma için kullanılan parametre değerleri Tablo 5.17'de verilmektedir.

Tablo 5.17. Simülasyon için kullanılan genetik algoritma parametreleri

Parametre	Değeri
Populasyon Büyüklüğü	60
Maksimum İşlem Sayısı	30
Çaprazlama oranı	%100
Mutasyon oranı	0.3

Aşağıda m-file için kullanılan genetik algoritmanın, algoritmik adımları ayrıntılı olarak açıklanmaktadır:

Adım 1. İkili sayı dizileri ile rastlantısal olarak ilk popülasyon oluşturulur.

Adım 2. Oluşan diziler, yukarıda açıklandığı üzere önce 2'li sisteme sonra onluk sisteme çevrilerek, sayısal verilere dönüştürülür. Kodlama ve dönüştürme işlemleri, Tablo 5.16'da ayrıntılı olarak verilmektedir. Sayısal verilere dönüştürülen popülasyon elemanları, simülasyon ortamına gönderilerek, sistemde oluşan hata değerleri belirlenir.

Adım 3. Simülasyondan gelen hata değerleri, uygunluk fonksiyonunda yerine konularak, uygunluk değerleri elde edilir. Uygunluk değerini minimum yapan popülasyon elemanları, yeni K1, K2 ve K3 değerleri olarak kullanılır. Uygunluk değerinin belirlenmesi için, denklem 5.7 seçilmiştir.

Adım 4. İlk oluşturulan popülasyon elemanları, çaprazlama işlemine tabi tutularak, yeni popülasyon oluşturulur. Adım 2 ve adım 3 kullanılarak yeni K1, K2 ve K3 belirlenir.

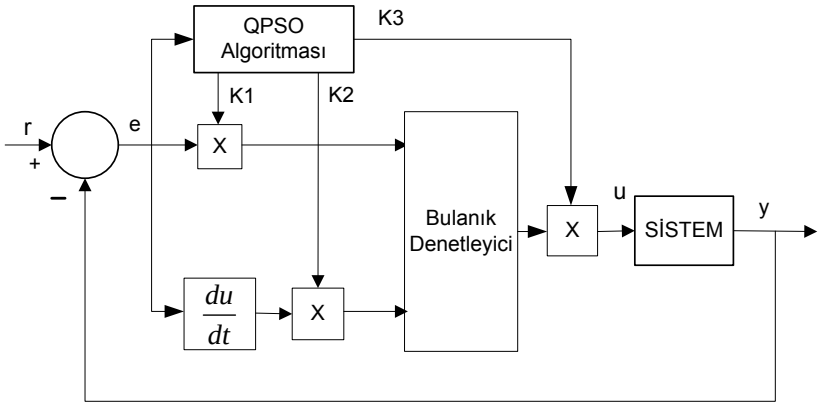
Adım 5. Popülasyona mutasyon uygulanarak, yeni popülasyon oluşturulur. Adım 2 ve adım 3 uygulanarak, yeni K1, K2 ve K3 değerleri belirlenir.

Adım 6. Yukarıdaki adımlarda bulunan en iyi uygunluk değerleri karşılaştırılır. Hangi adımdaki uygunluk değeri minimum ise, o adımda bulunan K1, K2 ve K3 değerleri sisteme gönderilir.

Adım 7. Maksimum işlem sayısına ulaşılana kadar bu adımlar uygulanır. Maksimum işlem sonunda, en uygun K1, K2 ve K3 değerleri belirlenir.

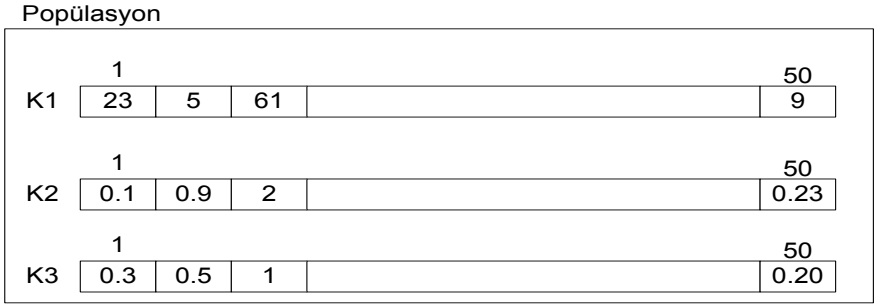
5.3.3. Kuantum Parçacık Sürü Optimizasyonu Çözümü

Bu bölümde, bulanık denetleyici parametrelerinin ayarlanması için, QPSO-bulanık denetleyici modeli önerilmiştir. Önerilen sistem için, Şekil 5.21'de simülink modeli verilmektedir. Algoritma, başlangıçta rastgele oluşturulan bir popülasyon ile başlamakta, oluşan popülasyon elemanları K1, K2 ve K3 değerlerini göstermektedir. Önerilen modelde, K1, K2 ve K3 için 0-4 aralığı kullanılmakta ve gerçek değerler ile kodlama yapılmaktadır. Aralık seçimi, uygulayıcının deneyimi ile belirlenmektedir. Şekil 5.22'de gösterildiği üzere, K1, K2 ve K3 değerleri QPSO algoritması ile bulunarak, simülink ortamına gönderilmektedir. Simülink ortamında elde edilen hata değerleri, algoritmaya gönderilerek, uygunluk değerlerinin bulunması sağlanmaktadır.



Şekil 5.22. Önerilen QPSO-bulanık denetleyici modeli

Şekil 5.23'de, K1, K2 ve K3 parametrelerinin, QPSO algoritması ile kodlanması verilmektedir. Şekil 5.23'te gösterildiği üzere, popülasyon büyüklüğü kadar K1, K2 ve K3 değerleri, rastlantısal olarak oluşturulmaktadır.



Şekil 5.23. Bulanık denetleyici ölçekleme faktörlerinin QPSO algoritması ile gösterimi

Bulanık denetleyici parametrelerinin kodlanması Tablo 5.18'de özet şeklinde verilmektedir.

Tablo 5.18. QPSO ile bulanık denetleyici parametrelerinin kodlanması

	QPSO ile kodlama	Kp ve Ki için minimum değer	Kp ve Ki için maksimum değer
K1	0, 12, 5, 7, 63, 55	0	4
K2	0.01, 0.06, 0.2, 2, 1	0	4
K3	0, 0.09, 1.7, 2, 0.01	0	4

Bulanık denetleyici parametrelerinin ayarlanması için, QPSO algoritması uygulanarak, matlab 7.0 ile m-file yazılmıştır. Simülasyon sonuçlarının bulunması için, Şekil 5.22'de verilen model, simülink ortamında oluşturulmuştur. Simülasyon işleminde, QPSO algoritması için kullanılan parametre değerleri Tablo 5.19'da verilmektedir.

Tablo 5.19. Simülasyon için kullanılan QPSO parametreleri

Parametre	Değeri
Populasyon Büyüklüğü	60
Maksimum İşlem Sayısı	30
c1, c2	0.1, 2
β	0.5

Aşağıda, m-file için kullanılan QPSO algoritmasının algoritmik adımları, ayrıntılı olarak açıklanmaktadır:

Adım 1: K1, K2 ve K3 için ilk popülasyon rastlantısal olarak oluşturulur.

Adım 2: Popülasyon elemanları, simülasyon ortamına gönderilerek, sistemde oluşan hata değerleri belirlenir.

Adım 3. Simülasyondan gelen hata değerleri, uygunluk fonksiyonunda yerine konularak, uygunluk değerleri elde edilir. Uygunluk değerini minimum yapan popülasyon elemanları, yeni K1, K2 ve K3 değerleri olarak kullanılır. Uygunluk değerinin belirlenmesi için, denklem 5.7 seçilmiştir.

Adım 4: Bulunan uygunluk değerleri, parçacıkların yerel konum değerlerini verir. Her bir parçacığa ait bu yerel değerler karşılaştırılır ve uygunluk değeri en iyi olan parçacığın konumu, popülasyonun yeni konumu olarak alınır.

Adım 5: Popülasyonun aritmetik ortalama değeri Mbest, Denklem 5.9 kullanılarak bulunur. Eğer, bulunan değer önceki değerlerden iyi ise, yeni değer Mbest olarak değiştirilir.

Adım 5: Popülasyonun en iyi hızı, Denklem 5.8 kullanılarak hesaplanır.

Adım 6: Parçacıkların yeni pozisyonu Denklem 5.10 kullanılarak güncellenir.

Adım 7: Maksimum işlem sayısına ulaşılana kadar 2, 3, 4, 5 ve 6 adımları tekrarlanır. Maksimum işlem sayısına ulaşıldığında, algoritma durdurulur. En iyi elemanlar seçilerek K1, K2 ve K3 değerleri için optimum çözüm elde edilir.

5.3.4. Simülasyon Sonuçları

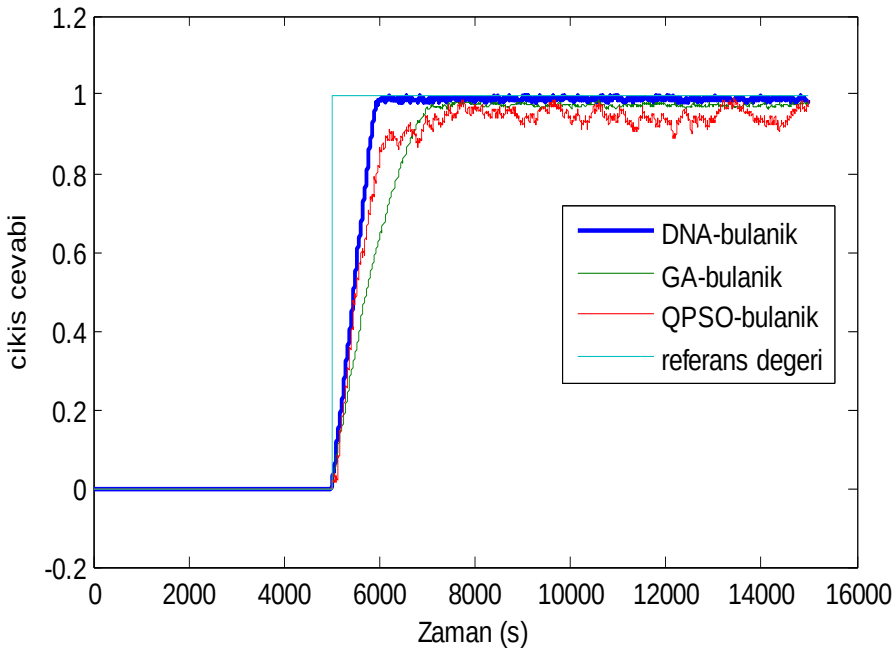
Önerilen yöntemin performans ölçümü için, Denklem 5.11 ile verilen bir DC motorun, pozisyon kontrolünü modelleyen transfer fonksiyonu, simülasyon sonuçları için kullanılmıştır. Yapılan uygulamada, bulanık denetleyici parametrelerinin optimizasyonu için, DNA hesaplama, genetik algoritma ve QPSO algoritması kullanılmıştır. Bulanık denetleyici parametrelerinin optimize edilmesinde, birçok uygunluk fonksiyonu kullanılmakla beraber, bu çalışmada, hatanın mutlak değerinin toplamı, uygunluk fonksiyonu olarak seçilmiştir. Çok küçük değerlerde hatalara dahi duyarlı olan bu fonksiyonun kullanımı ile üst aşım, yükselme ve oturma zamanlarının daha iyi sonuç vermesi hedeflenmiştir.

Yapılan matlab ve simülink çalışmasında, uygulamada kullanılan tüm algoritmalar için, popülasyon büyüklüğü 80, maksimum işlem sayısı 20 ve referans değeri 1 olarak alınmıştır. K1, K2 ve K3 değerlerinin tespiti için, denklem 5.7 uygunluk fonksiyonu olarak kullanılmıştır. Optimizasyon algoritmaları, çevrimiçi olarak sisteme uygulanmıştır. Rastlantısal olarak oluşturulan popülasyon elemanları, simülink ortamına gönderilerek, bulanık denetleyici elemanlarının optimizasyonu sağlanmıştır. Programın birçok kez çalıştırılması sonucu, sistemin ürettiği sonuçlar, Tablo5.20 ve Şekil 5.22'de verilmektedir.

Tablo 5.20. Bulanık denetleyicili sistem için simülasyon sonuçları

Algoritma	K1	K2	K3	Yerleşme Zamanı	Maksimum Aşım %
DNA-Bulanık Denetleyici	0.1250	0.2500	3.2500	0.1	%0
GA-Bulanık Denetleyici	0.0625	0	0.1250	0.2	%0
QPSO-Bulanık Denetleyici	2.3699	0.0236	0.0339	0.2	%0

Tablo 5.20 incelendiğinde, DNA hesaplama algoritması kullanılarak bulunan yerleşme zamanı değerinin, diğer algoritmalara göre daha iyi olduğu görülmektedir. Tablo 5.20'de verildiği üzere, DNA hesaplama algoritması ile yerleşme zamanı 0.1 sn iken, diğer algoritmalar ile 0.3 sn olmaktadır. Maksimum aşım, uygulanan tüm algoritmalarda yaklaşık ~%0 olarak bulunmaktadır. Diğer algoritmaların çalıştırılması ile bulunan sonuçlarda, karşılaştırma amaçlı Tablo 5.17'de verilmektedir.

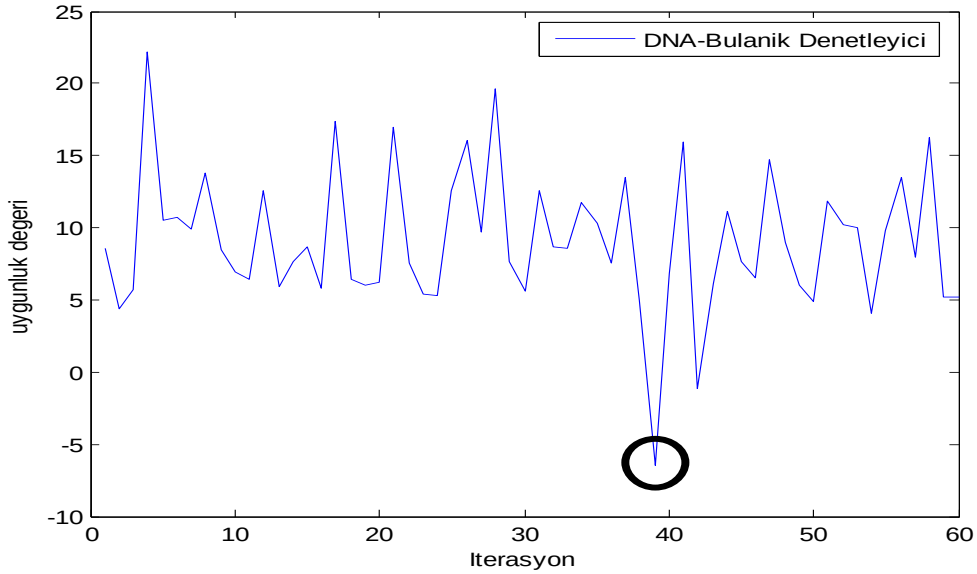


Şekil 5.24. Bulanık denetleyicili sistem için algoritma sonuçları

Sayısal verilerden görüleceği üzere, DNA hesaplama algoritması kullanıldığında, diğer algoritmalara göre daha kısa sürede yerleşme olmaktadır. Şekil 5.24, algoritmalar ile üretilen değerlerin simülink modeline gönderilmesi sonucu, sitemin oluşturduğu çıkış değerlerinin çizdirilmesi ile elde edilmiştir. Tablo 5.7 ile verilen yerleşme zamanı ve maksimum aşım Şekil 5.24 üzerinde net olarak görülmektedir.

DNA hesaplama algoritması ile kodlama yapılırken K1, K2 ve K3 değerleri 3 DNA bazı ile, 6 bit veri kullanılarak kodlanmıştır. Popülasyonda, ilk olarak 60 birey kullanılmış, her birey 18 bit veri ile ifade edilmiştir. Bu 18 bit verinin, ilk 6 biti K1 için, ikinci 6 biti K2 için ve diğer 6 biti ise K3 için kullanılmıştır. Her bir iterasyonda, popülasyonun %30'u kadar enzim ve virüs mutasyonu uygulanarak, 60*0.3=18 bireyin değişimi sağlanmış ve popülasyon yenilenmiştir.

DNA hesaplama algoritması, denetleyici parametrelerinin ayarlanmasında henüz çok fazla kullanılmamasına karşın, bu çalışma ile başarılı sonuçların elde edilebileceği gösterilmiştir. Popülasyon elemanlarına ait uygunluk değerlerinin değişimi, Şekil 5.25 ve Tablo 5.20'de verilmektedir. Tablo 5.21 ayrıntılı incelendiğinde, en iyi uygunluk değerini -6.5135 ile, popülasyonun 39. elemanı sağladığı görülmektedir. Bunun için, optimize edilmiş değerler K1=0.1250, K2=0.2500 ve K3=3.2500 olarak seçilmiştir.



Şekil 5.25. DNA hesaplama algoritması uygunluk fonksiyonu değerleri

Tablo 5.21 ve Şekil 5.25'te, çözüm için seçilen popülasyon elemanları ve uygunluk değerleri verilmektedir. Uygunluk değerlerinin, -10 ile 25 aralığında değiştiği görülmektedir. Uygunluk fonksiyonunu minimum yapan 39. popülasyon elemanı, çözüm olarak seçilmiştir.

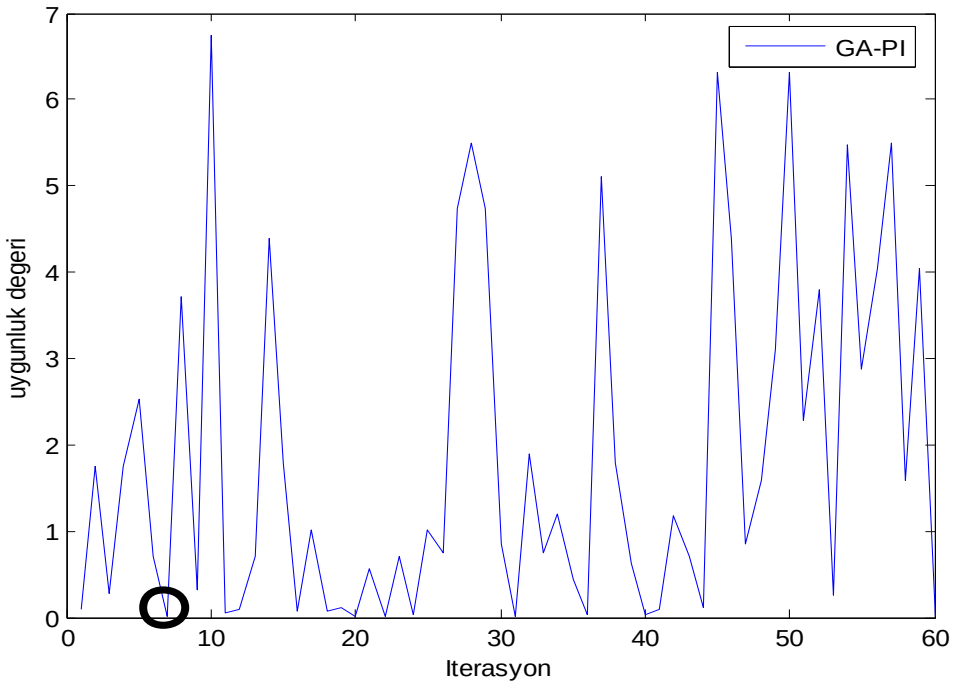
Tablo 5.21. DNA hesaplama ile elde edilen uygunluk, K1, K2 ve K3 değeri

iteras yon	uygunluk değeri	K1 değeri	K2 değeri	K3 değeri	iteras yon	uygunluk değeri	K1 değeri	K2 değeri	K3 değeri
1	8.5696	1.8125	1.1250	2.0000	31	12.5886	1.0625	2.1250	2.0625
2	4.4024	3.9375	0.8125	2.0000	32	8.7092	2.4375	2.9375	3.1250
3	5.6901	0.9375	0.9375	3.5625	33	8.5812	1.9375	2.0000	0.6875
4	22.2116	0.1875	3.8750	0	34	11.7119	0.7500	1.4375	2.6250
5	10.5673	1.4375	2.3125	3.7500	35	10.2849	1.4375	1.5000	1.4375
6	10.7334	1.8750	3.8750	3.4375	36	7.5465	2.6250	1.9375	3.5625
7	9.9336	2.0000	3.5000	2.6250	37	13.5153	1.0000	3.6250	0.3750
8	13.7920	0.8750	1.3125	0.5000	38	4.9881	3.2500	1.0000	3.8750
9	8.4831	2.7500	3.8125	3.0000	**39**	**-6.5135**	**0.1250**	**0.2500**	**3.2500**
10	6.9508	3.4375	3.8125	2.4375	40	6.8571	2.1875	0.3750	0.5000
11	6.3766	2.8750	1.8750	1.1250	41	15.9159	0.6250	2.4375	0.6875
12	12.5371	1.0625	1.0000	0.1250	42	-1.1459	0.8750	0	2.6250
13	5.9595	2.3125	0	1.1875	43	6.1676	1.6250	0.7500	3.0625
14	7.6432	3.0000	2.8750	3.4375	44	11.0970	1.0625	1.5000	2.6250
15	8.7086	1.7500	0.7500	0.1250	45	7.6995	1.1875	1.4375	3.9375
16	5.8356	3.0625	1.3125	1.8125	46	6.4794	3.3750	2.4375	2.7500
17	17.3901	0.3750	2.1250	1.9375	47	14.7499	0.7500	1.8125	0.6250
18	6.4393	3.0625	2.4375	1.4375	48	8.9331	2.1250	2.2500	2.8750

Tablo 5.21'in devamı

19	5.9822	3.8750	2.2500	3.6875	49	5.9651	3.8750	3.1875	2.3125
20	6.2122	2.5000	0.5625	1.5625	50	4.9215	3.5625	1.0000	0.6250
21	17.0068	0.3125	2.7500	2.8750	51	11.8514	1.3125	2.9375	3.9375
22	7.5477	2.6250	2.4375	0.3125	52	10.2583	1.3125	0.8125	1.3125
23	5.3849	3.2500	0.9375	0.2500	53	9.9822	1.4375	0.3750	0.1250
24	5.3330	3.2500	0.9375	3.0000	54	4.1091	3.5625	0	1.6875
25	12.5216	1.2500	3.7500	1.1250	55	9.7716	0.6875	0.7500	2.0625
26	16.0072	0.5000	1.8750	2.0000	56	13.4385	1.0625	3.5625	1.0000
27	9.6934	0.7500	1.2500	2.9375	57	7.9671	2.8750	3.0000	3.5000
28	19.6609	0.3125	3.8125	0.4375	58	16.2683	0.6875	2.1250	0.3125
29	7.6473	2.5000	2.3125	1.8125	59	5.2456	3.3125	0.9375	0.3750
30	5.5978	3.8750	2.7500	1.6875	60	5.1742	3.0000	0.2500	0.5000

Genetik algoritma ile kodlama yapılırken, K1, K2 ve K3 değerleri 6 bit veri kullanılarak kodlanmıştır. Popülasyonda, ilk olarak 60 birey kullanılmış, her birey 18 bit veri ile ifade edilmiştir. Bu 18 bit verinin, ilk 6 biti K1, ikini 6 bit K2 için ve diğer 6 biti ise Ki için kullanılmıştır. Her bir iterasyonda, popülasyonun %30'u kadar enzim ve virüs mutasyonu uygulanarak, 60*0.3=18 bireyin değişimi sağlanmış ve popülasyon yenilenmiştir.



Şekil 5.26. Genetik algoritma uygunluk fonksiyonu değerleri

Genetik algoritma popülasyonuna ait uygunluk değerleri, Şekil 5.26 ve Tablo 5.22'de verilmiştir. Şekil 5.26 incelendiğinde, en iyi uygunluk değerini, 0.0000 ile popülasyonun 7.

elemanı sağlamıştır. 7. Elemanın kullanımı sonucu oluşan değerler, optimize edilmiş değerler olup, K1=0.0625, K2=0 ve K3=0.1250 olarak seçilmiştir. Diğer elemanların uygunluk değerleri incelendiğinde, daha büyük veriler elde edildiği, Tablo 5.22'de gösterilmektedir.

Tablo 5.22. GA ile elde edilen uygunluk, K1, K2 ve K3 değeri

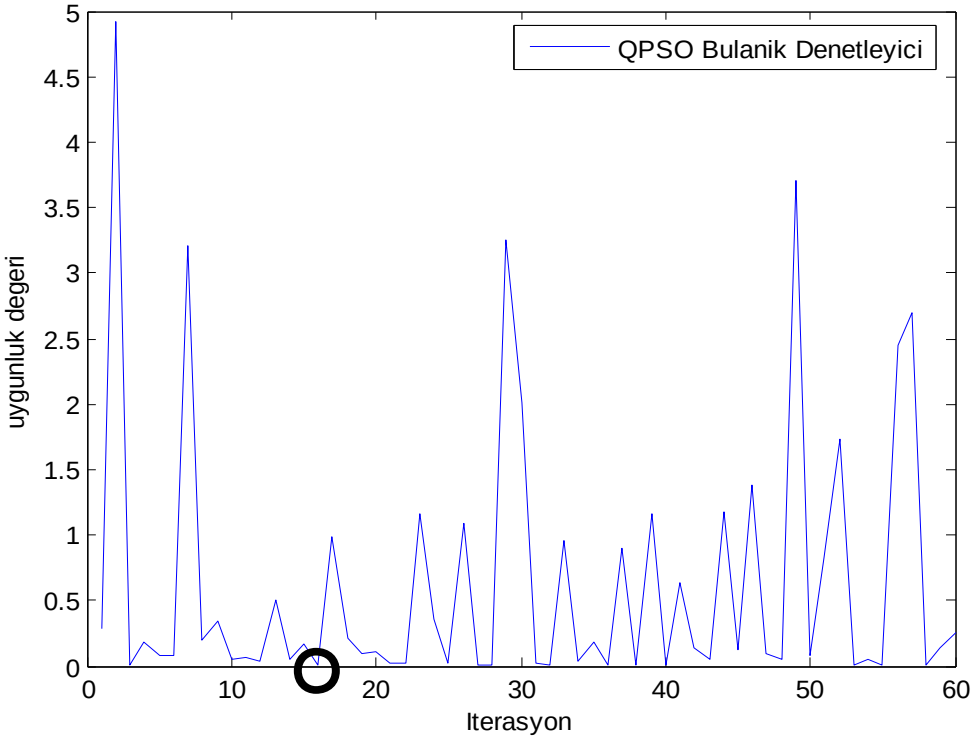
iteras yon	uygunluk değeri	K1 değeri	K2 değeri	K3 değeri	iteras yon	uygunluk değeri	K1 değeri	K2 değeri	K3 değeri
1	0.0834	0.7500	0.4375	0.2500	31	0.0070	1.5000	0.3125	0.0625
2	1.7577	0.6250	1.7500	1.6875	32	1.8903	0.8750	0	1.0625
3	0.2703	0.6875	0.5625	0.5625	33	0.7509	0.3125	1.3125	1.5625
4	1.7577	0.6250	1.7500	1.6875	34	1.1921	0.4375	1.1250	1.6875
5	2.5329	1.2500	1.7500	1.1875	35	0.4413	0.2500	1.4375	1.4375
6	0.7016	1.3750	0.6250	0.6250	36	0.0209	0.7500	1.3750	0.1250
7	**0.0000**	**0.0625**	**0**	**0.1250**	37	5.1150	1.2500	1.0625	1.6875
8	3.7117	1.8750	0	1.4375	38	1.7962	1.6250	1.7500	1.0000
9	0.3205	0.8750	0.7500	0.4375	39	0.6218	0.3750	1.1875	1.3125
10	6.7428	1.1250	0.3750	1.9375	40	0.0281	1.4375	1.0000	0.1250
11	0.0534	0.6875	1.9375	0.2500	41	0.0834	0.6875	1.0000	0.3125
12	0.0973	0.3125	1.7500	0.5625	42	1.1858	1.5000	0.1875	0.8125
13	0.7016	1.8750	1.5000	0.6250	43	0.7016	1.5000	0.5000	0.6250
14	4.3853	1.7500	0	1.5625	44	0.1123	1.5625	0	0.2500
15	1.79620.	1.9375	0.9375	1.0000	45	6.3148	1.3750	0.4375	1.8750
16	0631	1.0000	1.7500	0.1875	46	4.3853	1.8750	0.5000	1.5625
17	1.0104	1.2500	1.3125	0.7500	47	0.8490	1.8125	1.4375	0.6875
18	0.0631	1.0625	0.6250	0.1875	48	1.5787	1.5625	0.2500	0.9375
19	0.1123	1.2500	1.6250	0.2500	49	3.0943	1.8125	1.5625	1.3125
20	0.0065	0.8750	0.6875	0.0625	50	6.3148	1.8750	0.5000	1.8750
21	0.5683	1.8750	0.0625	0.5625	51	2.2733	1.2500	1.1250	1.1250
22	0.0040	0.0625	0.4375	1.3750	52	3.8013	0.7500	1.5000	1.6875
23	0.7016	1.5625	0.0625	0.6250	53	0.2526	1.0000	0.8125	0.3750
24	0.0281	1.8750	0.9375	0.1250	54	5.4738	0.8125	1.1250	1.8750
25	1.0104	1.3125	1.9375	0.7500	55	2.8845	0.8750	1.9375	1.3125
26	0.7509	0.6875	0.5625	0.9375	56	4.0415	1.7500	1.9375	1.5000
27	4.7431	1.0000	0.3125	1.6250	57	5.5009	1.8750	1.9375	1.7500
28	5.5009	1.3750	0.0625	1.7500	58	1.5787	1.7500	1.1875	0.9375
29	4.7431	1.1250	1.3125	1.6250	59	4.0415	1.9375	0.4375	1.5000
30	0.8490	1.8125	0.6875	0.6875	60	0.0014	0.0625	0.8125	0.8125

QPSO ile hesaplama yapılırken popülasyonda, K1 için 20, K2 için 20 ve Ki için 20 olmak üzere toplam 60 birey kullanılmıştır. QPSO algoritması ile elde edilen popülasyona ait uygunluk değerleri, Şekil 5.27 ve Tablo 5.23'de verilmiştir. Burada en iyi uygunluk değeri 16. elemana ait 0.0000 değeri olduğundan, optimize edilmiş K1=2.3699, K2=0.0236 ve K3=0.0339 değerleri olarak kabul edilmiştir.

Tablo 5.23. QPSO ile elde edilen uygunluk, K1, K2 ve K3 değeri

iterasyon	uygunluk değeri	K1 değeri	K2 değeri	K3 değeri	iterasyon	uygunluk değeri	K1 değeri	K2 değeri	K3 değeri
1	0.2839	3.0757	1.7807	4.7520	31	0.0263	0.6608	0.4666	2.5913
2	4.9245	8.2963	7.6781	4.1397	32	0.0026	0.0621	0.6285	3.8929
3	0.0004	0.0627	0.5658	1.5444	33	0.9625	0.0547	6.2972	4.3040
4	0.1814	1.4098	1.1426	4.2299	34	0.0414	0.0652	0.3279	3.9494
5	0.0811	0.0765	0.5867	3.4329	35	0.1838	3.0757	1.3950	3.9319
6	0.0749	1.4098	0.8241	3.1713	36	0.0004	0.0587	0.1139	1.4527
7	3.2019	8.2963	7.1198	4.5723	37	0.9032	8.2963	4.3531	3.1989
8	0.1958	3.5313	0.6988	3.4812	38	0.0116	0.0587	1.5358	3.9095
9	0.3456	3.9000	1.8437	4.7431	39	1.1587	0.0547	7.4863	5.4826
10	0.0558	2.6148	0.9730	2.6666	40	0.0021	0.0638	0.2669	1.4707
11	0.0700	2.6148	1.0845	2.9074	41	0.6331	3.9263	0.0716	3.9411
12	0.0325	0.0587	1.9196	4.7304	42	0.1317	2.2217	1.2821	3.6182
13	0.5031	3.8886	0.3193	3.8546	43	0.0564	0.0706	0.4662	3.4938
14	0.0465	0.0750	0.4778	2.8173	44	1.1818	8.2963	4.8394	3.4696
15	0.1739	3.9263	0.0573	2.1065	45	0.1235	3.9000	1.3096	3.2677
16	**0.0000**	**2.3699**	**0.0236**	**0.0339**	46	1.3881	8.2963	4.6482	2.9684
17	0.9891	8.2963	4.6025	3.4084	47	0.0938	0.0547	3.2963	2.2302
18	0.2184	4.1519	0.4413	2.6895	48	0.0584	0.0765	0.6035	3.2438
19	0.0989	0.0745	0.4823	3.8100	49	3.6996	8.2963	7.2949	4.4385
20	0.1043	1.4098	1.0228	3.5348	50	0.0751	0.0765	0.6308	3.6028
21	0.0185	2.6148	0.6320	1.7737	51	0.8314	8.2963	3.2437	1.8365
22	0.0289	3.0757	0.7456	2.0893	52	1.7387	0.0547	7.6591	4.7100
23	1.1668	0.0547	8.0273	6.1826	53	0.0106	0.0707	0.4407	1.7354
24	0.3646	8.2963	2.8796	2.1942	54	0.0456	0.0750	0.5435	2.8841
25	0.0148	0.0707	0.4567	2.0922	55	0.0122	0.0707	0.5031	2.0198
26	1.0841	0.0547	7.1652	5.2035	56	2.4469	3.2229	3.1585	4.1776
27	0.0004	0.0618	0.2617	1.0049	57	2.6988	8.2963	7.4883	5.4591
28	0.0047	0.6608	0.2425	1.3763	58	0.0116	0.6608	0.3861	1.9767
29	3.2498	8.2963	7.1009	4.5109	59	0.1459	3.0757	1.0452	3.5943
30	2.0130	8.2963	6.1495	4.2986	60	0.2516	3.0007	0.8920	4.2176

Simülasyon sonuçlarından görüldüğü gibi, önerilen DNA hesaplama tabanlı algoritma, diğer algoritmalara göre, daha performanslı sonuçların elde edilmesini sağlamıştır. Literatürde verilen, geleneksel sayısal DNA hesaplama algoritması için; çaprazlama, mutasyon, enzim ve virüs mutasyonu, kaydırma, tersten yazma ve karıştırma adımları uygulanmaktadır. Her uygulamada, bu sayılan adımlardan birkaçı veya tamamı kullanılabilmektedir. Bu çalışmada ise, kullanılan DNA hesaplama algoritmasının basitleştirilmesi için; çaprazlama, enzim ve virüs mutasyonu adımları kullanılmıştır.



Şekil 5.27. QPSO algoritması uygunluk fonksiyonu değerleri

5.4. Bölüm Değerlendirmesi

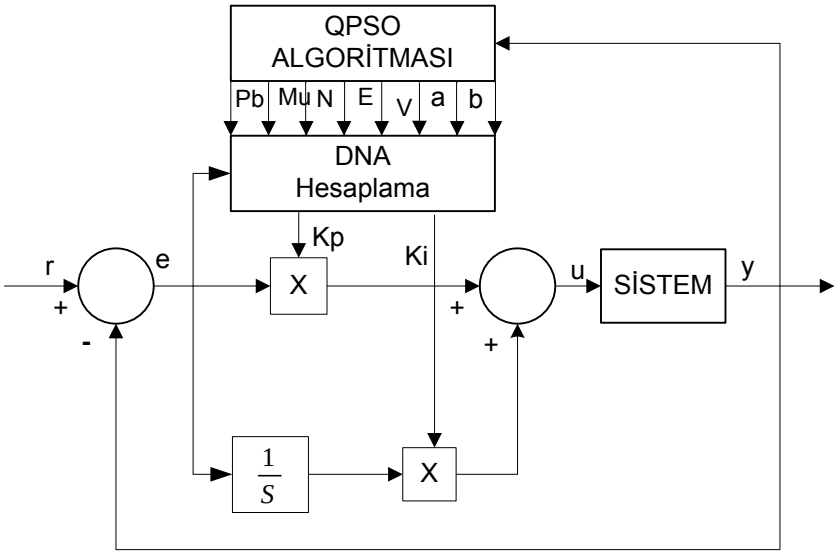
Literatürde, PI ve bulanık denetleyici parametrelerinin ayarlanması için, kullanılan algoritmaların çoğu çözelti ortamında gerçekleştirilmiş olup, uygulaması zor ve maliyetlidir. Bazı sayısal ortam algoritmaları ise, DNA dizilerini sayısal verilere dönüştürürken, ikili sayı sistemini kullanmıştır. İkili sayı sitemine çevrilirken, 3 baz uzunluğunda DNA dizisi 6 bit veri ile ifade edilmektedir. Örneğin, ATG – 001110 gibi. Bu tez çalışmasında, dizilerin sayısal verilere dönüştürülmesinde, 4'lü sayı sistemi kullanılmıştır. Örneğin, ATG – 031 gibi. Özellikle, değişken sayısı çok olan problemlerde, ikili sayı sistemi, uzun ikili bit dizilerinin oluşmasına neden olup, hesaplamayı zorlaştırmakta, işlem süresini uzatmakta ve problemin kontrolünü zorlaştırmaktadır. Bu tezde yapılan uygulama ile, belirtilen risklerin giderilmesi sağlanmıştır. Yapılan çalışma ile, DNA hesaplamanın sayısal ortamda uygulanabilirliği gösterilmiştir.

6. QPSO TABANLI UYARLAMALI DNA HESAPLAMA

DNA Hesaplama, son yıllarda optimizasyon problemlerinin çözümünde sıklıkla kullanılmaktadır. Bu uygulamalarda, DNA hesaplamanın veri saklama ve hızlı veri işleme özellikleri öne çıkmaktadır. DNA hesaplama uygulanırken, öncelikle DNA molekülleri ile rastlantısal olarak bir çözüm alanı oluşturulmakta ve bu alanda optimum nokta aranmaktadır. Problemin gerçek çözümünün bulunmasında, çözüm alanının iyi oluşturulması gerekmektedir. DNA ile hesaplama yapılırken, simülasyon için birçok parametre kullanılmaktadır. DNA hesaplamaya ait bu parametrelerin iyi seçilmesi, problemlerin optimum çözümünün elde edilmesinde önemli role sahiptir [46]. DNA hesaplamanın iyi sonuçlar üretmesi için gerekli olan bu parametreler: Popülasyon büyüklüğü, maksimum işlem sayısı, çaprazlama oranı, enzim ve virüs oranı olarak sayılabilir.

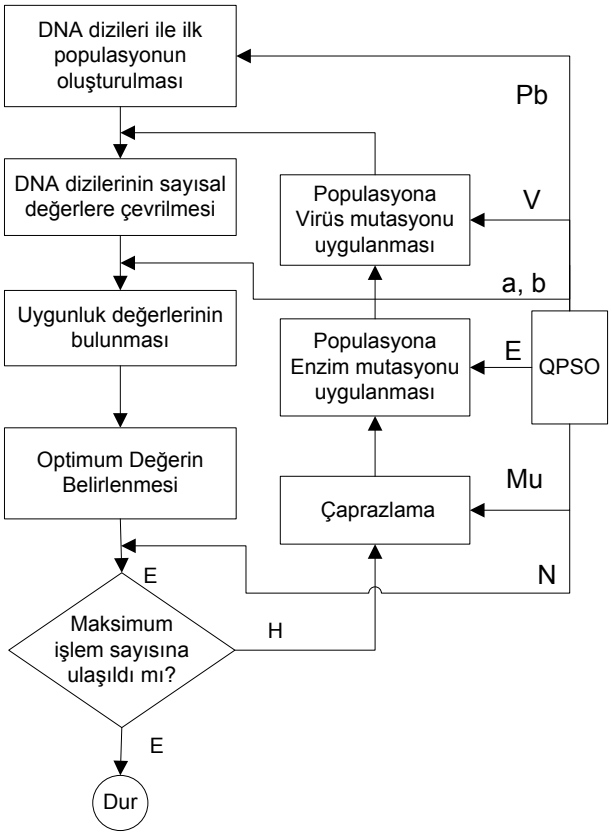
6.1. Uyarlamalı DNA ile Kodlama

Bu bölümde, uyarlamalı DNA hesaplama algoritması ile populasyon_boyutu (Pb), maksimum_islem_sayısı (N), çaprazlama_oranı (Mu), enzim_oranı (E), virüs_oranı (V), uygunluk fonksiyonu için kullanılan a ve b parametrelerinin aktif hale getirilmesi sağlanmaktadır. Yapılan uygulamada, algoritmanın her bir iterasyonunda popülasyonun elde ettiği uygunluk değerleri dikkate alınarak, parametrelerin optimum değerlerinin bulunması için çalışılmaktadır. Uygunluk fonksiyonu dikkate alınarak parametre değerleri artırılıp azaltılmakta ve optimum çözüme ulaşıldığında elde edilen değerler, optimize edilmiş parametre değerleri olarak belirlenmektedir. Başlangıçta sabit olan bu parametreler, dinamik hale getirilerek algoritmadan daha iyi bir verim alınması amaçlanmaktadır. Şekil 6.1'de önerilen yöntemin modellemesi gösterilmektedir. Şekil üzerinde görüldüğü gibi DNA hesaplama parametre değerleri QPSO ile bulunarak DNA hesaplama algoritmasına gönderilmekte ve her iterasyonda değerlerin güncellenerek kullanılması sağlanmaktadır.



Şekil 6.1. Önerilen QPSO tabanlı uyarlamalıDNA – PI modeli

Şekil 6.2'de QPSO tabanlı uyarlamalı DNA hesaplama akış diyagramı gösterilmektedir.



Şekil 6.2. Önerilen QPSO tabanlı uyarlamalı DNA algoritması

DNA hesaplamanın uyarlanabilir duruma getirilmesi için kullanılan parametre değerleri aşağıda açıklanmaktadır;

Populasyon Büyüklüğü (Pb): Popülasyon büyüklüğü, problemin çözümüne katkı veren en önemli parametrelerden biridir. Bu parametrenin değeri belirlenirken, problemin büyüklüğü ve karmaşık yapısı dikkate alınır. Popülasyon büyüklüğü artırıldığında daha geniş bir çözüm alanı oluşturulmasına karşılık çözüm zamanı uzamaktadır. Ayrıca, çözüm alanının büyütülmesi gereksiz zaman kayıplarına neden olmaktadır. Bu nedenle uygulamalarda optimum değerin seçilmesi gerekmektedir. Populasyon büyüklüğünün ayarlanmasında denklem 6.1 kullanılmıştır.

$$Pop\_boyutu=(max(f)-min(f))/(f(n)-f(1))+15 \quad (6.1)$$

Denklem 6.1'de verilen f değişkeni algoritma için kullanılan uygunluk fonksiyonudur. f(n) popülasyonun n. elemanına ait uygunluk değerini, f(1) ise 1. elemana ait uygunluk değerini vermektedir. Max(f), popülasyona ait maksimum değeri, min(f) ise minimum değeri ifade etmektedir. Denklem 6.1 içerisinde verilen 15 sayısı taban değer olup populasyon boyutunun sıfır olması ihtimalini yok etmek için kullanılmakta, deneme ve yanılma yöntemi ile belirlenmektedir.

Maksimum işlem sayısı (N):Maksimum işlem sayısı, DNA hesaplama algoritmasında kullanılan döngünün işleme sayısını vermektedir. Bu parametrenin değeri artırıldığında, döngülerin tamamlanma süresi uzamakta ve yapılacak işlem sayısı artmaktadır. Özellikle popülasyonu oluşturan eleman sayısının ve problemde kullanılan değişken sayısının fazla olması durumunda, algoritmanın tamamlanma süresi hızla artmakta zaman ve ekonomik kayıplara neden olmaktadır. Maksimum işlem sayısının en az düzeyde tutulması ise algoritmanın verimliliğini düşürmektedir. Bütün bu durumları dikkate alarak en iyi değerlerin belirlenmesi gerekmektedir. Bu çalışmada, maksimum işlem sayısının belirlenmesi için Denklem 6.2 kullanılmıştır.

$$Maksimum\_islem\_sayısı=(f(n)+f(1))*Pb+5 \quad (6.2)$$

Denklem 6.2 içerisinde verilen 5 sayısı taban değer olup, maksimum işlem sayısının sıfır olması ihtimalini yok etmek için kullanılmakta, deneme ve yanılma yöntemi ile belirlenmektedir.

Çaprazlama Oranı (Mu):Çaprazlama ile, var olan popülasyon elemanları kullanılarak yeni elemanlar oluşturulmaktadır. Çaprazlama oranı belirlenirken, çaprazlanacak elemanların sayısı ve seçimi dikkate alınmaktadır. Çaprazlama oranı yüksek tutulduğunda, seçilecek eleman sayısı artmakta ve popülasyonun değişimi sağlanmaktadır. Ancak oluşacak yeni elemanların ebeveyn elemanlara yakın değerler olma olasılığı vardır. Ayrıca çaprazlanacak eleman sayısının çok olması, algoritmanın tamamlanma süresini uzatmaktadır. Bu çalışmada çaprazlama oranını bulmak için Denklem 6.3 kullanılmıştır.

caprazlama_orani=Pop_boyutu*0.8 (6.3)

Denklem 6.3 ile, populasyon her bir iterasyonda %80 oranında çaprazlamaya tabi tutulmaktadır.

Enzim (E) ve Virüs (V) oranı: Bu çalışmada kullanılan en önemli iki parametre, enzim ve virüs mutasyonudur. Enzim mutasyonu, popülâsyondan belirli oranda elemanın silinmesini sağlarken, virüs mutasyonu ise popülâsyona belirli oranda yeni elemanın eklenmesini sağlamaktadır. Bu iki parametre diğerlerinden farklı olarak, sadece DNA hesapla ile kullanılmaktadır. Uyarlamalı DNA hesaplama uygulanırken enzim ve virüs oranı iyi seçilmelidir. Çünkü enzim mutasyonu doğru zamanda uygulanmadığında, uygunluk değeri iyi olan elemanların kaybedilmesine neden olabilmektedir. Virüs mutasyonu uygulanırken, algoritmanın gereksiz yerde çözüm alanı oluşturmaması için dikkatli olmak gerekmektedir. Yapılan çalışmada, enzim ve virüs oranını belirlemek için Denklem 6.4 kullanılmıştır.

e, v= Pop_boyutu*0.3 (6.4)

Denklem 6.4 ile populasyona, her bir iterasyonda %30 oranında enzim ve virüs mutasyonu uygulanmaktadır. Yukarıda verilen denklemlerin seçiminde deneme yanılma yöntemi ile bulunan sonuçlar dikkate alınmış, sistem birçok kez çalıştırılarak elde edilen değerler irdelenip denklemlere son şekli verilmiştir.

DNA hesaplama algoritmik adımları ve açıklaması:

Aşağıda m-file için kullanılan DNA hesaplama algoritmasının, algoritmik adımları ayrıntılı olarak açıklanmaktadır.

Adım 1: DNA dizileri ile rastlantısal olarak ilk popülasyon oluşturulur.

Adım 2: Oluşan diziler Tablo 6.1'de açıklandığı üzere önce 4'lü sisteme sonra onluk sisteme çevrilerek sayısal verilere dönüştürülür. Kodlama ve dönüştürme işlemleri Tablo 6.1'de ayrıntılı olarak verilmektedir. Sayısal verilere dönüştürülen popülasyon elemanları simülasyon ortamına gönderilerek sistemde oluşan hata değerleri belirlenir.

Adım 3: Simülasyondan gelen hata değerleri uygunluk fonksiyonunda yerine konularak uygunluk değerleri elde edilir. Uygunluk değerini minimum yapan popülasyon elemanları, yeni Kp ve Ki değerleri olarak kullanılır. Uygunluk değerinin belirlenmesi için Denklem 6.5 seçilmiştir.

$$f(Kp,Ki)=sum(abs(e)) \tag{6.5}$$

*Adım 4:*UyarlamalıDNA hesaplama algoritması ve döngü için kullanılacak parametre değerleri QPSO ile bulunarak sisteme gönderilir. QPSO uygunluk fonksiyonu için Denklem 6.6kullanılmaktadır. Denklemlerin seçiminde deneme yanılma yöntemi ile bulunan sonuçlar dikkate alınmış, sistem birçok kez çalıştırılarak elde edilen değerler irdelenip denklemlere son şekli verilmiştir.

$$f\_qpso=a*sum(e^2)+b*max(yout) \tag{6.6}$$

$$a=sum(f\_qpso)*c1 \tag{6.7}$$

$$b=average(f\_qpso)*c2 \tag{6.8}$$

Adım 5: İlk oluşturulan popülasyon elemanları çaprazlama işlemine tabi tutularak yeni popülasyon oluşturulur. Adım 2 ve adım 3 kullanılarak yeni Kp ve Ki değerleri belirlenir.

Adım 6: Popülasyona enzim ve virüs mutasyonu uygulanarak yeni popülasyon oluşturulur. Adım 2, 3 ve 4 uygulanarak yeni Kp ve Ki değerleri belirlenir.

Adım 7: Yukarıdaki adımlarda bulunan en iyi uygunluk değerleri karşılaştırılır. Hangi adımdaki uygunluk değeri minimum ise o adımda bulunan Kp ve Ki değerleri sisteme gönderilir.

Adım 8: Maksimum işlem sayısına ulaşılana kadar bu adımlar uygulanır. Maksimum işlem sonunda en uygun Kp ve Ki değerleri belirlenir.

6.2. Simülasyon Sonuçları

Şekil 6.1'de görüldüğü üzere sistem, QPSO tabanlı olarak DNA hesaplama ile modellenmekte ve parametrelerin ayarlanması için DNA dizileri kullanılmaktadır. Önerilen modelde, Kp ve Ki değişkenleri 3 baz uzunluğunda DNA dizileri ile kodlanmaktadır. Değişkenlerin alacağı değerler A, G, C ve T bazları kullanılarak rastlantısal olarak oluşturulmakta, sıra ile 0, 1, 2 ve 3 değerleri kullanılarak sayısal verilere dönüştürülüp sisteme gönderilmektedir. DNA hesaplama ile yapılan kodlama örneği Tablo 6.1'de verilmiştir.

Tablo 6.1. PI parametrelerinin uyarlamalı DNA hesaplama ile kodlanması

	DNA Kod Sistemi	4'lü Kod Sistemi	2'li Kod Sistemi	Onluk sisteme çeviri	Kp ve Ki için Minimum Değer	Kp ve Ki için Maksimum Değer
	A, G, C, T	0, 1, 2, 3	00, 01, 10, 11			
Kp	AGT, TAA, TTT	013, 300, 333	000111, 110000, 111111	0*16+1*4+3*1=7 3*16+0*4+0*1=48 3*16+3*4+3*1=63	0	63
Ki	GTT, CCA, AGC	133, 220, 012	011111, 101000, 000110	1*1+3*1/4+3*1/16=1.9375 2*1+2*1/4*0*1/16=2.5 0*1+1*1/4+2*1/16=0.375	0	4

Önerilen yöntemin performans ölçümü için, Denklem 6.9 ile verilen bir DC motorun pozisyon kontrolünü modelleyen transfer fonksiyonu, simülasyon sonuçları için kullanılmıştır. PI parametrelerinin ayarlanması için matlab7.0 ile DNA hesaplama algoritmasını uygulayarak m-file yazılmaktadır. Simülasyon sonuçlarının bulunması için Şekil 6.1'de verilen model simülink ortamında oluşturulmaktadır.

$$G(s) = \frac{2.2}{8.96e-6s^3 + 7.27e-3s^2 + 0.945s} \tag{6.9}$$

DNA hesaplama için kullanılan parametre değerleri Tablo 6.2'de verilmektedir. Bu değerler DNA hesaplamanın ilk kullanımı için uygulanmakta, QPSO ile bulunan değerler geldiğinde değişmektedir.

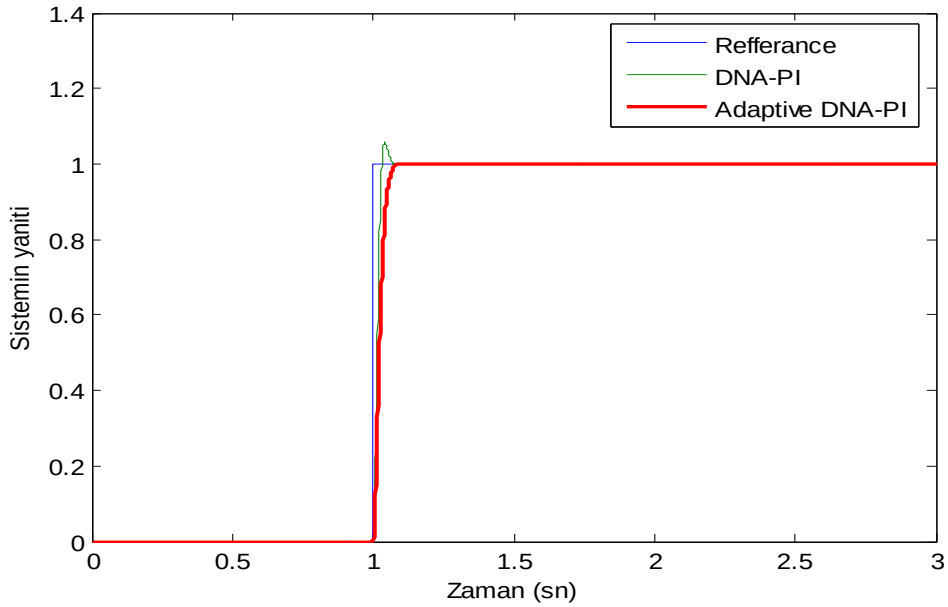
Tablo 6.2. Simülasyon için kullanılan DNA hesaplama parametreleri

Parametre	Değeri
Populasyon Büyüklüğü	60
Maksimum İşlem Sayısı	30
Çaprazlama oranı	%100
Enzim ve Virüs Oranı	0.3

Yapılan uygulamada, PI parametrelerinin optimizasyonu için, QPSO tabanlı uyarlamalı DNA hesaplama algoritması kullanılmıştır. PI parametrelerinin optimize edilmesinde birçok uygunluk fonksiyonu kullanılmakla beraber, bu çalışmada hatanın mutlak değerinin toplamı, uygunluk fonksiyonu olarak seçilmiştir. Çok küçük değerlerde hatalara dahi duyarlı olan bu fonksiyonun kullanımı ile üst aşım, yükselme ve oturma zamanlarının daha iyi sonuç vermesi hedeflenmiştir. Matlab ve simülink çalışmasında, uygulamada kullanılan popülasyon büyüklüğü 80, maksimum işlem sayısı 20 ve referans değeri 1 olarak alınmıştır. Kp ve Ki değerlerinin tespiti için, Denklem 6.5 uygunluk fonksiyonu olarak kullanılmıştır. Programın birçok kez çalıştırılması sonucu sistemin ürettiği sonuçlar Tablo 6.3 ve Şekil 6.3'de verilmektedir. Tablo 6.3'de verildiği üzere uyarlamalıDNA hesaplama algoritması ile Kp 17, Ki 0.4375, yerleşme zamanı 0.08 sn ve maksimum aşım yaklaşık %0 olarak bulunmuştur. Şekil 6.3'te uyarlamalı DNA hesaplama ile bulunan sonuçların DNA hesaplama ile karşılaştırılması verilmektedir. Uyarlamalı hale getirilen DNA hesaplama algoritması ile bulunan maksimum aşım değeri yaklaşık %0 iken, DNA hesaplama ile bulunan maksimum aşım değeri yaklaşık %14 olarak hesaplanmaktadır. Bu sonuçlar uyarlamalı DNA hesaplamanın daha iyi bir sonuç elde ettiğini göstermektedir.

Tablo 6.3. DNA ve uyarlamalı DNA simülasyon sonuçları

Algoritma	Kp değeri	Ki değeri	Yerleşme Zamanı	Maksimum Aşım %
DNA-PI	30	0.1250	0.08	14
Uyarlamalı-DNA-PI	17	0.4375	0.08	~0



Şekil 6.3. Uyarlamalı sistem için simülasyon sonuçları

Tablo 6.4'te QPSO algoritması ile uyarlamalı hale getirilen DNA hesaplama parametreleri ve DNA hesaplama parametreleri verilmektedir. DNA hesaplama algoritması ile kodlama yapılırken,Kp ve Ki değerleri 3 DNA bazı ile 6 bit veri kullanılarak kodlanmıştır. Popülasyonda ilk olarak 80 birey kullanılmış, her birey 12 bit veri ile ifade edilmiştir. Bu 12 bit verinin ilk 6 biti Kp için diğer 6 biti ise Ki için kullanılmıştır. Her bir iterasyonda, popülasyonun %30'u kadar enzim ve virüs mutasyonu uygulanarak 80*0.3=24 bireyin değişimi sağlanmış ve popülasyon yenilenmiştir. Uyarlamalı DNA ile elde edilen yeni popülasyon büyüklüğü 60 olarak bulunmuştur.

Tablo 6.4. DNA ve uyarlamalı DNA parametre değerleri

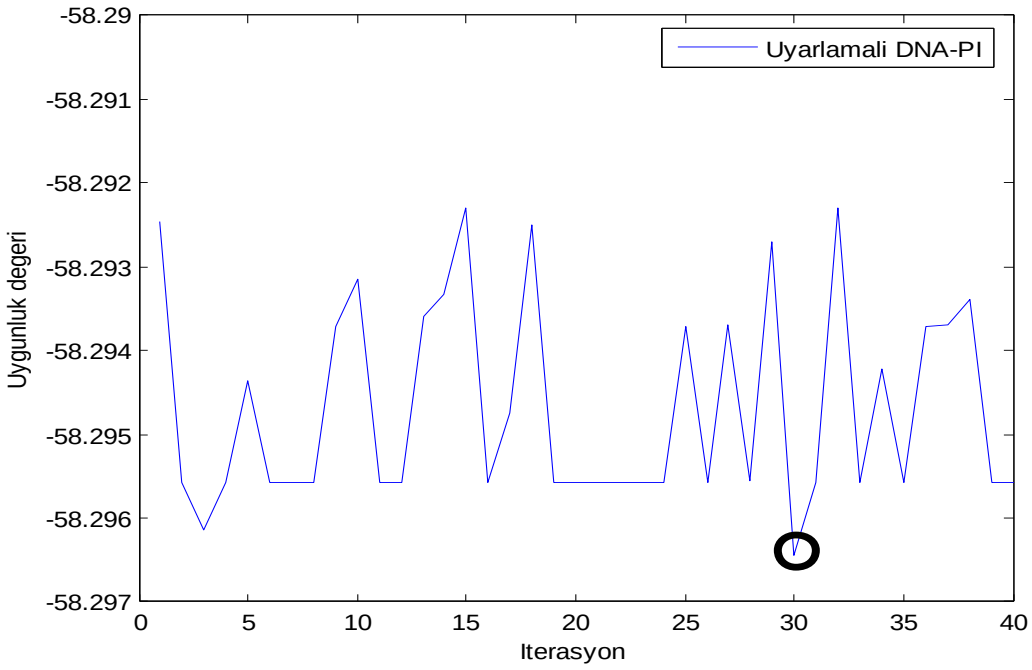
Parametre	DNA Hesaplama Değeri	Uyarlamalı DNA Hesaplama Değeri
Populasyon Büyüklüğü	80	60
Maksimum İşlem Sayısı	20	40
Çaprazlama oranı	%100	%32
Enzim ve Virüs Oranı	%30	%12

Popülasyon elemanlarına ait uygunluk değerlerinin değişimi Şekil 6.4 ve Tablo 6.5'te verilmektedir. Tablo 6.5 ayrıntılı incelendiğinde en iyi uygunluk değerini -58.2964 değeri

ile popülasyonun 30. elemanının sağladığı görülmektedir. Bunun için optimize edilmiş ve uyarlanabilir hale getirilmiş değerler Kp=17 ve Ki=0.4375 olarak seçilmiştir.

Tablo 6.5. UyarlamalıDNA hesaplama ile elde edilen uygunluk, Kp ve Ki değerleri

iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri	iterasyon	uygunluk değeri	Kp değerleri	Ki değerleri
1	-58.2925	36	3.5625	21	-58.2956	58	0.5625
2	-58.2956	58	0.5625	22	-58.2956	58	0.5625
3	-58.2961	52	1.5625	23	-58.2956	58	0.5625
4	-58.2956	58	0.5625	24	-58.2956	58	0.5625
5	-58.2944	19	3.0000	25	-58.2937	12	0.2500
6	-58.2956	58	0.5625	26	-58.2956	58	0.5625
7	-58.2956	58	0.5625	27	-58.2937	15	0.5625
8	-58.2956	58	0.5625	28	-58.2956	58	1.1250
9	-58.2937	5	0.0625	29	-58.2927	2	2.7500
10	-58.2931	48	3.5000	**30**	**-58.2964**	**17**	**0.4375**
11	-58.2956	58	0.5625	31	-58.2956	58	0.5625
12	-58.2956	58	0.5625	32	-58.2923	60	3.6875
13	-58.2936	8	1.1250	33	-58.2956	58	0.5625
14	-58.2933	6	2.5625	34	-58.2942	38	3.0000
15	-58.2923	60	3.5000	35	-58.2956	58	0.5625
16	-58.2956	58	0.5625	36	-58.2937	62	1.0625
17	-58.2948	44	1.4375	37	-58.2937	40	2.6875
18	-58.2925	36	2.0000	38	-58.2934	3	1.2500
19	-58.2956	58	0.5625	39	-58.2956	58	0.5625
20	-58.2956	58	0.5625	40	-58.2956	58	0.5625



Şekil 6.4. Uyarlamalı DNA hesaplama algoritması uygunluk fonksiyonu değerleri

DNA hesaplama parametrelerinin bulunması için QPSO algoritması kullanılmıştır. Çünkü QPSO algoritması, belirli bir noktaya odaklanma özelliğine sahiptir. Çalışmada, QPSO algoritması hataya bağlı uygunluk fonksiyonu ile DNA hesaplama parametrelerini elde etmiştir. Simülasyon sonuçları, QPSO tabanlı uyarlamalı DNA hesaplama algoritmasının daha performanslı olduğunu göstermektedir.

6.3. Bölüm Değerlendirmesi

Bu bölümde, sayısal DNA hesaplama algoritması, QPSO algoritması ile uyarlanabilir duruma getirilmiştir. Yapılan uygulama ile algoritma dinamik bir yapıya dönüştürülerek, parametre değerlerinin sistem tarafından belirlenmesi sağlanmıştır.Özellikle, QPSO algoritmasının belirli bir noktaya odaklanma özelliği, parametre değerlerinin belirlenmesinde önemli rol oynamıştır.

Yapılan uygulama ile literatürdeki DNA hesaplama algoritmalarında, uygulayıcının deneme yanılma yoluyla ve birçok kez deneyerek elde ettiği değerler, daha kolay ve daha doğru olarak bulunmaktadır.

6. SONUÇLAR

Bu çalışmada, NP problemlerden gezgin satıcı ve sırt çantası problemi ile PI ve bulanık denetleyici parametrelerinin ayarlanması için, sayısal DNA hesaplama algoritmasının kullanılabileceği gösterilmiştir. DNA hesaplamanın iki uygulama türünden biri olan sayısal DNA hesaplama algoritmasının NP problemler ve denetleyici parametrelerinin optimizasyonu için uygulaması yapılmıştır. Yapılan çalışmada, DNA hesaplamanın sağladığı en belirgin avantaj, uygulanan mutasyon yöntemi ile daha geniş bir alanda çözüm kümesinin oluşmasıdır. Bununla yerel optimum noktalara bağlı kalmaksızın genel arama yeteneği kazanılmaktadır. Literatürde kullanılan DNA hesaplama algoritmaları daha kompleks olup bu çalışmada basitleştirilmiştir. Önerilen DNA hesaplama algoritmasının, geleneksel DNA hesaplama algoritmasına göre uygulanabilirliği daha kolay olup daha hızlı sonuç elde edilmektedir. Bu algoritmanın sistemlerin optimizasyonu için etkili bir araç olduğu kanıtlanmıştır. Bununla birlikte, Sayısal DNA hesaplamanın, çözelti ortamında uygulanan DNA hesaplama kadar etkili olamayacağı açıktır. Çünkü sayısal ortamda yapılan hesaplamada, DNA moleküllerinin gerçek özellikleri hesaplama algoritmasına tam olarak uygulanamamaktadır.

Bu tez çalışmasında kullanılan uygulamalarda, örnek problemler için değişken sayıları sınırlı tutulmuştur. DNA hesaplamanın asıl avantajı, değişken sayısı arttığında görülmektedir. Fakat sayısal ortamda değişken sayısının artırılması, problemin çözüm süresini uzatmakta ve çözümün bulunması zor olmaktadır. Yapılan çalışma ile, DNA hesaplamanın sayısal ortamlarda da uygulanabileceğinin gösterilmesi amaçlanmıştır.

Tez çalışmasında öncelikle DNA hesaplama, genetik algoritma ve QPSO algoritması NP problemlerden gezgin satıcı ve sırt çantası problemine uygulanmıştır. Uygulama sonucu aynı değerler elde edilmekle birlikte, DNA hesaplamanın daha kısa sürede sonuca ulaştığı görülmüştür.

İkinci bölümde yapılan uygulamada, DNA hesaplama ve diğer algoritmalar PI ve bulanık denetleyici parametrelerinin ayarlanması için kullanılmıştır. Simülasyon sonuçları irdelendiğinde, DNA hesaplamanın diğer algoritmalara göre daha iyi bir performans sağladığı görülmüştür.

Son uygulamada, sayısal DNA hesaplama, literatürdeki DNA algoritmasından farklı olarak, QPSO algoritması ile uyarlanabilir hale getirilmiştir. Yapılan çalışmada, DNA hesaplama parametreleri QPSO ile bulunarak sistem dinamik bir yapıya kavuşturulmuştur. Simülasyon sonuçlarından görüleceği üzere, literatürde kullanılan DNA hesaplama algoritmalarına göre önerilen yöntem çok daha iyi sonuçlar üretmiştir.

Bütün bu sonuçlardan, DNA hesaplama algoritmasının NP problemlerin optimizasyonu ve denetleyici parametrelerinin ayarlanmasında daha etkili olduğu sonucuna varılmaktadır. Bu alandaki çalışmalardan gelecekteki beklentimiz, DNA moleküllerinden oluşan DNA bilgisayarların hızla geliştirilmesi ve DNA hesaplamanın daha etkin bir şekilde kullanımının sağlanmasıdır. Daha sonra yapılacak çalışmada bu algoritma geliştirilerek performansı artırılacaktır. Ayrıca burada uygulanan problemlerin dışında diğer problemlere de uygulanarak performansı ölçülecektir.

KAYNAKLAR

[1] Feynman, R.P., 1961. Miniaturization, D.H.Gilbert (Ed.), New York, Reinhold, p. 282-296.

[2] Adleman, L.M., 1994. Molecular computation of solutions to combinational problems, Science, 266(4), p. 1021-1025.

[3] Muhammad, M.S., Ibrahim, Z., Ono, O. and Khalid, M., 2005. Direct-Proportional Length-Based DNA Computing Implementation for Elevator Scheduling Problem, TENCON 2005 IEEE Region 10, 21-24 Nov., p. 1 – 5.

[4] Li, Y., 2007. Traveling Salesman Problem Based on DNA Computing, Third International Conference on Natural Computation, 24-27 Aug., vol. 4, p. 28 – 34.

[5] Han, A., 2006. DNA Computing Model for the 0/1 Knapsack Problem, Hybrid Intelligent Systems HIS '06, Sixth International Conference, 26 Dec., p. 18 – 18.

[6] Xikui, L. and Yan, L., 2009. DNA computing for Traveling Salesman problem, Bioinformatics and Biomedical Engineering, ICBBE 3rd International Conference, 11-13 June, p. 1 – 4.

[7] Han, A. and Zhu, D., 2006. DNA Computing Model for the Minimum Spanning Tree Problem, Symbolic and Numeric Algorithms for Scientific Computing, SYNASC '06, Eighth International Symposium, Sept., p. 372 – 377.

[8] Zhou, K., Tong, X. and Xu, J., 2006. The Improvement on Algorithm of DNA Computing on 0-1 Planning Problem, Proceedings of the fifth international conferance on Machine Learning and Cybernetics Dalian, 13-16 Aug., p. 4282 – 4286.

[9] Yang, C.N., Chao, C.H., Cheng, H.P. and Lin, C.H., 2006. A MEMS Based DNA Computer for Solving SAT Problems, Nano/Micro Engineered and Molecular Systems NEMS '06, 1st IEEE International Conference, 18-21 January, p. 172 – 177.

[10] Sridhar, R., Balasubramaniam, S., 2011. GIS integrated DNA computing for solving Travelling Salesman Problem, Computers & Informatics (ISCI), p. 402 – 406.

[11] Ibershoff, J., Jaromczyk, J.W. and Noort, D., 2007. Making the SAT Decision Based on a DNA Computation, Evolutionary Computation, IEEE Congress, 25-28 Sept., p. 1835 – 1842.

[12] Huang, X., Yin, Z., Zhi, L. and Hu, J., 2009. Using Implicit Enumeration to Solve 0-1 Planning Problem Based on DNA Computing, Artificial Intelligence and Computational Intelligence, AICI '09 International Conference, 7-8 Nov., vol. 1, p. 629 – 633.

[13] Shin, S.Y., Zhang, B.T., and Jun, S.S., 1999. Solving travelling salesman problems using molecular programming, Evolutionary Computation, Proceedings of the Congress, 6-9 July, Washington.

[14] Lipton, R.J, 1995. DNA solutions of hard computational problems, Science, vol. 268, p. 542-545.

[15] Lipton, R.J., 1995. Using DNA to solve NP-complete problems, Science, vol. 268(4), p. 542-545.

[16] Ouyang, Q., Kaplan, P., Liu, S., and Libchaber, A., 1997. DNA solution of the maximal Clique Problem, Science, vol. 278, p. 446-449.

[17] Tomohiro, Y., Takeshi, F., and Yoshiki, U., 1996. DNA Coding Method and a Mechanism of Development for Acquisition of Fuzzy Control Rules, Proceedings of IEEE International Conference on Fuzzy Systems, San Antonio, TX, USA, 8-11 Sept., vol. 3, p. 2194-2200.

[18] Yoshinobu, T., Aoi, Y., Tanizawa, K., and Iwasaki, H., 1998. Ligation errors in DNA computing, in Fourth International Meeting on DNA Based Computers, Philadelphia, June , p. 245–246.

[19] Maley, C. C., 1998. DNA Computation: Theory,Practice, and Prospects, Evolutionary Computation, Vol. 6, p. 201-229.

[20] Wood, D., Chen, J., Antipov, E., Lemieux, B., and Cedeño, W., 1999. A DNA Implementation of the Max 1s Problem, Proceedings of Genetic and Evolutionary Computation, Orlando, FL, USA, p. 1-7.

[21] Chen, J., Antipov, E., Lemieux, B., Cedeno, W., and Wood, D. H., 1999. DNA Computing Implementing Genetic Algorithms, Proceeding of DIMACS Workshop on Evolution as Computation, Princeton, NJ, USA, p. 39-49.

[22] Ding, Y., and Ren, L., 2000. DNA Genetic Algorithm for Design of the Generalized Membership-Type Takagi-Sugeno Fuzzy Control System, Proceedings of IEEE International Conference on Systems, man, and Cybernetics, 8-11 Oct., vol. 5, p. 3862-3867.

[23] Yanfeng, W., Guangzhao, C. and Zhiwu, W., 2005. Research on DNA computer with coprocessor organizational model, High-Performance Computing in Asia-Pacific Region, Proceedings Eighth International Conference, 1-1 July, vol. 6, p. 552.

[24] Xu, J., Qiang, X., Yang, Y., Wang, B., Yang, D., Luo, L., Pan, L., Wang S., 2011. An Unenumerative DNA Computing Model for Vertex Coloring Problem, NanoBioscience, Vol. 10 , p. 94 – 98.

[25] Qiu, Z., Lu, M., 2003. A new approach to advance the DNA computing, applied soft computing, vol. 3, p. 177-189.

[26] Forbes, N., 2000. Biological inspired computing, Computing in Science & Engineering, Nov. - Dec, Washington, vol. 2, pp. 83–87.

[27] Ibrahim, Z., Tsuboi, Y. and Ono, O., 2006. Hybridization-ligation versus parallel overlap assembly: an experimental comparison of initial pool generation for direct-proportional length-based DNA computing, NanoBioscience, IEEE Transactions, vol. 5, p. 103 – 109.

[28] Boneh, D., Dunworth, C., Sgall, J., and Lipton, R.J., 1996. Making DNA computers error resistant, in Second Annual Meeting on DNA Based Computers, Princeton, June, p. 102–110.

[29] Cui, G., Quin, L., Wang, Y. and Zhang, X., 2007. Information Security Technology Based on DNA Computing, Anti-counterfeiting, Security, Identification, IEEE International Workshop, 16-18 April, p. 288 – 291.

[30] Clelland, C.T., Risca, V. and Bancroft, C., 1999. Hiding messages in DNA microdots Nature, vol. 399, p. 533–534.

[31] Shin, S.Y., Lee, I.H., D., Kim and B.T., Zhang, 2005. Multiobjective Evolutionary Optimization of DNA Sequences for Reliable DNA Computing, Evolutionary Computation, IEEE Transactions on Vol. 9, p. 143 – 158.

[32] Henkel, C.V., 2005. Experimental DNA Computing, Doktora Tezi, Leiden Üniversitesi, Gravenhage.

[33] Qiu, Z.F., 2003. Advance the DNA computing, Doktora Tezi, Texas A&M University, Texas.

[34] Baum, E.B., 1995. Building an associative memory vastly larger than the brain, Science 268, 583–585.

[35] Reif, J.H., LaBean, T.H., Pirrung, M., Rana, V.S., Guo, B., Kingsford C., and Wickham, G.S., 2002. Experimental construction of very large scale DNA databases with associative search capability, N. Jonoska & N.C. Seeman (eds.) DNA computing, 7th international meeting on DNA-based computers. Springer-Verlag, Berlin Heidelberg, p. 231–247.

[36] Lee, J.Y., Shin, S.Y., Augh, S.J., Park T.H., and Zhang, B.T., 2003. Temperature gradient-based DNA computing for graph problems with weighted edges, M. Hagiya & A. Ohuchi (eds.) DNA computing, 8th international workshop on DNA-based computers. Springer-Verlag, Berlin Heidelberg, p. 73–84.

[37] Head,T., Rozenberg, G., Bladergroen, S. and Breek C., 2000. Computing with DNA by operating on Plasmids, BioSystems, vol. 57, p.87-93.

[38] Kwon, O.H., Wang, K.Y., Kim, J.Y., Park, J., Chung, D.J. and Lee, C.H., 2007. DNA Inspired Digital Signal Pattern Matching Algorithm, Frontiers in the Convergence of Bioscience and Information Technologies FBIT 2007, 11-13 Oct., p. 753 – 756.

[39] Fujiwara, A., Matsumoto, K. and Chen, W., 2003. Addressable procedures for logic and arithmetic operations with DNA strands, Parallel and Distributed Processing Symposium Proceedings International, 22-26 April, p. 8.

[40] Emel, G.G. and Taşkın, Ç., 2002. Genetik algoritmalar ve uygulama alanları, Uludağ Üniversitesi iktisadi ve idari bilimler fakültesi dergisi, cilt 19, sayı 1, s. 129-152.

[41] Ma, Y., Zhang, and S., Wang, J., 2010. The comparisons and selections of optimization methods in engineering, international conference on Machine Vision and Human-machine interface, 24-25 April, p. 650-652.

[42] Subaşı, Ö., 2007. Öz ayarlamalı ve sıçramasız geçişli kontrol sistemi tasarımı, Yüksek lisans tezi, İstanbul Teknik Üniversitesi, İstanbul.

[43] Xi, M., Sun, J., and Xu, W., 2007. Parameter optimization of PID controller based on Quantum-Behaved Particle Swarm Optimization Algorithm, Complex systems and applications-modelling, control and simulations, vol 14 (S2), p. 603-607.

[44] Duman, E. and Akın, E., 2006. Yapay bağışık sistemi ile bulanık yaklaştırıcı kural tabanı optimizasyonu, Elektrik-elektronik-bilgisayar sempozyumu, Uludağ Üniversitesi-Eleco'06, Bursa.

[45] Lin, C.L., Jan, H.Y and Hwang, T.S., 2004. Structure Variable PID Control Design Based on DNA Coding Method , Industrial Electronics, International Symposium, 4-7 May, vol. 1, p. 423 – 428.

[46] Lin, C.L., Jan, H.Y. and Huang, T.H., 2004. Self-organizing PID Control Design Based on DNA Computing Method, Proceedings of the International Conference, 2-4 Sept.,vol. 1, p. 568 – 573.

[47] Jan, H.Y., Lin, C.L., Hwang, T.S., 2006. Self-Organized PID control design using DNA computing approach, Journal of Chinese Institute of engineers, vol.29, no.2, p. 251-261.

[48] Kim, J.J., Lee, J.J., 2007. PID controller design using double helix structured DNA algorithms with a recovery function, 12th International Symposium on Artifical Life and Robotics, 25-27,Japan.

[49] Zhang, Z., Shi, X. and Liu, J., 2008. A method to encrypt information with DNA computing, Bio-Inspired Computing: Theories and Applications, 3rd International Conference, 28 Sept-1 Oct., p. 155 – 160.

[50] Steele, G. and Stojkovic, V., 2004. Agent-Oriented Approach to DNA Computing, Computational Systems Bioinformatics Conference CSB Proceedings, 16-19 Aug., p. 546 – 551.

[51] Liu, X. and Liu, H., 2009. Spatial Cluster Analysis Based on Evolutionary DNA Computing Technique, Bioinformatics and Biomedical Engineering ICBBE 3rd International Conference, 11-13 June, p. 1 – 4.

[52] Çiğdem, U., and Karaköse, M., 2011. PI ve bulanık denetleyici parametrelerinin ayarlanmasında DNA hesaplamanın kullanılması, Otomatik kontrol ulusal toplantısı, DEU, İzmir, 14-16 Eylül, S. 665-670.

[53] Hongyan, Z. and Xiyu, L., 2009. A General Object Oriented Description for DNA Computing Technique, Information Technology and Computer Science ITCS International Conference, 25-26 July, vol. 1, p. 3 – 6.

[54] Jiejun, W. and Chuan-pei, X., 2009. Test generation for combinational circuits Based on DNA computing, Electronic Measurement & Instruments, ICEMI 9th International Conference, 16-19 Aug., vol. 4, p. 650-653.

[55] Huang, Y. and He, L., 2011. DNA computing research progress and application, Computer Science & Education (ICCSE), 6th International Conference, 3-5 Aug., p. 232 – 235.

[56] Watada, J. and Bakar, R., 2008. DNA computing and its applications, ISDA '08, Eighth International Conference, 26-28 Nov., vol. 2, p. 288 – 294.

[57] Watanabe, S., Tsuboi, Y., Ibrahim, Z., Yamamoto, T. and Ono, O., 2004. Adaptive DNA computing algorithm by using PCR and restriction enzyme, Cybernetics and Intelligent Systems, Conference, 1-3 Dec., vol. 1, p. 262 – 267.

[58] Peng, X., Vadakkepat, P. and Lee, T.H., 2001. DNA coded GA for the rule base optimization of a fuzzy logic controller, Evolutionary Computation, Proceedings of the Congress, 27-30 May., vol. 2, p. 1191 – 1196.

[59] Wang, Y.F., Niu, Y. and Cui, G.Z., 2008. An efficient genetic algorithm based on the cultural algorithm applied to DNA codewords design, Bio-Inspired Computing: Theories and Applications, 3rd International Conference, 28 Sept.-1 Oct., p. 103 – 108.

[60] Xuncai, Z., Ying, N., Yanfeng, W., 2011. DNA Computing in Microreactors a Solution to the Minimum Vertex Cover Problem, Bio-Inspired Computing: Theories and Applications (BIC-TA), 27-29 Sept., p. 236 – 240.

[61] Wang, L., Yang, Y.X., Wang, A. and Ma, J.L., 2009. A DNA Algorithm of a Kind of Circular Forbidden Permutation Problem, Networks Security, Wireless Communications and Trusted Computing, NSWCTC '09. International Conference, 25-26 April, vol. 2, p. 13 – 16.

[62] Tsuboi, Y., Ibrahim, Z. and Ono, O., 2004. Problem-solving method with semantic net based on DNA computing in artificial intelligence, Control Conference 5th Asian, 20-23 July, vol. 1, p. 653 – 658.

[63] Minggang, D., Ning, W. and Jili, T., 2009. DNA Computing in Control Systems: A survey, Control and Decision Conference, CCDC '09, Chinese, 17-19 June, p. 4942 – 4946.

[64] Sekhar, R. , 2001. DNA Computing-Graph Algorithms, Department of Mathematics, Indian Institute of Technology,Indian.

[65] Mitra, S., Das, R., Hayashi Y., 2011. Genetic Networks and soft computing, IEEE/acm transactıons on computatıonal biology and bioinformatics, Jan.-Feb., vol. 8, p. 94-107.

[66] Amos, M., Paun, G., Rozenberg, G. and Salomaa, A., 2002. Topics in the theory of DNA computing, Theoretical Computer Science, vol. 287 (2002), p. 3 – 38.

[67] Jiao, H., Zhong, Y., Zhang, L. and Li, P., 2011. Unsupervised remote sensing image classification using an artificial DNA computing, Natural Computation (ICNC), vol. 3, p. 1341 – 1345.

[68] Liu, W., Sun, S. and Guo, Y., 2009. A DNA Computing Model of Perceptron, Circuits, Communications and Systems PACCS '09. Pacific-Asia Conference, 16-17 May, p. 726 – 729.

[69] Ibrahim, Z., Kurniawan, T.B., Khalid, M. and Ono, O., 2007. A DNA Sequence Design for Direct-Proportional Length-Based DNA Computing using DNASequenceGenerator, Innovative Computing, Information and Control, ICICIC '07, Second International Conference, 5-7 Sept., 364 – 364.

[70] Cui, G., Li, C., Li, H. and Li, X., 2009. DNA Computing and Its Application to Information Security Field, Natural Computation, ICNC '09, Fifth International Conference, 14-16 Aug., vol. 6,p. 148 – 152.

[71] Yin, Z., Yang, J., Yang Y. and Ma, Y.,2007. DNA Computing Model of the 0-1 Programming Problem, International Journal of Algebra, Vol. 1, no. 2, s. 71 – 79.

[72] Darehmiraki, M. and Nehi, H.M., 2007. Molecular solution to the 0–1 knapsack problem based on DNA computing, Applied Mathematics and Computation, 20 Sept., vol. 187 (2007), p. 1033–1037.

[73] Yağsan O., 2006. Bir operatörsüz vinç için PID ve genetik algoritma temelli minimum salınımlı konum kontrolü, Yüksek lisans tezi, Yıldız Teknik Üniversitesi, İstanbul.

[74] Saaid, M.F.M., Ibrahim, Z., Khalid, M. and Yahya, A., 2008. Alternative Fuzzy C-Means Clustering for DNA Computing Readout Method Implemented on DNA Engine Opticon 2 System, Signal Image Technology and Internet Based Systems, International Conference, Nov. 30- Dec. 3, p. 498 – 503.

[75] Rajaee, N., Aoyagi, H., Yabe, K., Kon, Y. and Ono, O., 2008. A new approach for Boolean matrix multiplication with DNA computing, Soft Computing in Industrial Applications Conference, 25-27 June, p. 44 – 48.

[76] Tsaftaris, S.A., Katsaggelos, A.K., Pappas, T.N. and Papoutsakis, E.T., 2004. How can DNA computing be applied to digital signal processing?, Signal Processing Magazine IEEE, Nov., vol. 21, p. 57 – 61.

[77] Saaid, M.F.M., Ibrahim, Z., Khalid, M. and Sarmin, N.H., 2009. Automation of DNA Computing Readout Method Implemented on LightCycler System, Modelling & Simulation, AMS '09, Third Asia International Conference, 25-29 May, p. 252 – 257.

[78] Song, T., Wang, S. and Wang, X., 2008. The Design of Reversible Gate and Reversible Sequential Circuit based on DNA Computing, Intelligent System and Knowledge Engineering, ISKE 2008, 3rd International Conference, 17-19 Nov., vol. 1, p. 114 – 118.

[79] Tsaftaris, S.A., Katsaggelos, A.K., Pappas, T.N. and Papaoutsakis, E.T., 2004. DNA computing from a signal processing viewpoint, Signal Processing Magazine IEEE, Sept., vol. 21, s. 100 – 106.

[80] Watada, J. and Bakar, R., 2009. Computational Cluster Validation in DNA-Based Computation, Intelligent Signal Processing, International Symposium, 26-28 Aug., p. 187 – 192.

[81] Chen, Z., Qu, H., Lu, M. And Zhu, H., 2004. A Probabilistic Parameterized Algorithm for Vertex Cover in Sticker Model, Parallel and Distributed Processing Symposium,Proceedings 18th International, 26-30 April, p. 166.

[82] Wang, Z. and Shao, Z., 2007. A Branch and Cut Algorithm for DNA Encoding, Bio-Inspired Computing: Theories and Applications, Second International Conference, 14-17 Sept., p. 249 – 253.

EKLER

PI Denetleyici için Çevrimiçi Parametre Ayarlamasını Sağlayan DNA Hesaplama Algoritması

```
n=80; % populasyon büyüklüğü
maxiter=20; % maksimum işlem sayısı
m=3; % DNA dizi uzunlukları belirleniyor.
for i=1:n
      pop(i,:)=randseq(m);% Kp için populasyon oluşturuluyor.
end
for i=1:n
      pop1(i,:)=randseq(m+1); % Ki için populasyon oluşturuluyor.
end
  %DNA dizileri sayıya çevriliyor. A:0,G:1,C:2,T:3
for i=1:n
for j=1:m+1
if pop1(i,j)=='A'
           pop1_sayi(i,j)=0;
elseif pop1(i,j)=='G'
           pop1_sayi(i,j)=1;
elseif pop1(i,j)=='C'
           pop1_sayi(i,j)=2;
elseif pop1(i,j)=='T'
           pop1_sayi(i,j)=3;
end
end
end
for i=1:n
for j=1:m
if pop(i,j)=='A'
pop_sayi(i,j)=0;
elseif pop(i,j)=='G'
pop_sayi(i,j)=1;
elseif pop(i,j)=='C'
pop_sayi(i,j)=2;
elseif pop(i,j)=='T'
pop_sayi(i,j)=3;
end
end
end
f_dna=zeros(n,1); %%%%%%%%%% Populasyon 10'lu sisteme çevriliyor.
for i=1:n

pop_ondalik(i,1)=pop_sayi(i,1)*16+pop_sayi(i,2)*4+pop_sayi(i,3)*1;%
pop1_ondalik(i,1)=pop1_sayi(i,1)*1+pop1_sayi(i,2)*1/4+pop1_sayi(i,3)*1/16+pop1_sayi(
i,4)*1/64;
```

```
populasyon_ondalik(1,i)=pop_ondalik(i,1);
populasyon_ondalik(2,i)=pop1_ondalik(i,1);
end
for i=1:n
f_dna(i)=tracklsq1(populasyon_ondalik(:,i)); % Fonksiyonun uygunluk değeri
hesaplanıyor.
end
uygunluk_deger=min(f_dna);
for i=1:n
ifuygunluk_deger==f_dna(i)
          k1=pop_ondalik(i,1);
          k2=pop1_ondalik(i,1);
end
end
 % Program için döngü başlatılıyor.
   f1=zeros(n,1);
iter=1;
while iter<=maxiter  % Maksimum işlem sayısı için döngü kullanılıyor.
iter=iter+1;
    % Çaprazlama ile yeni populasyon oluşturuluyor.
w=randperm(m);
pop_capraz(:,1)=pop(:,w(1));
pop_capraz(:,2)=pop(:,w(2));
pop_capraz(:,3)=pop(:,w(3));
w=randperm(m+1);
    pop1_capraz(:,1)=pop1(:,w(1));
    pop1_capraz(:,2)=pop1(:,w(2));
    pop1_capraz(:,3)=pop1(:,w(3));
    pop1_capraz(:,4)=pop1(:,w(4));
   %DNA dizileri sayıya çevriliyor. A:0,G:1,C:2,T:3
for i=1:n
for j=1:m+1
if pop1_capraz(i,j)=='A'
          pop1_capraz_sayi(i,j)=0;
elseif pop1_capraz(i,j)=='G'
          pop1_capraz_sayi(i,j)=1;
elseif pop1_capraz(i,j)=='C'
          pop1_capraz_sayi(i,j)=2;
elseif pop1_capraz(i,j)=='T'
          pop1_capraz_sayi(i,j)=3;
end
end
end
for i=1:n
for j=1:m
ifpop_capraz(i,j)=='A'
pop_capraz_sayi(i,j)=0;
elseifpop_capraz(i,j)=='G'
pop_capraz_sayi(i,j)=1;
```

```
elseifpop_capraz(i,j)=='C'
pop_capraz_sayi(i,j)=2;
elseifpop_capraz(i,j)=='T'
pop_capraz_sayi(i,j)=3;
end
end
end

f_capraz_dna=zeros(n,1); %%%%%%%%%%%% Populasyon 10'lu sisteme çevriliyor.

for i=1:n
pop_capraz_ondalik(i,1)=pop_capraz_sayi(i,1)*16+pop_capraz_sayi(i,2)*4+pop_capraz_s
ayi(i,3)*1;%
pop1_capraz_ondalik(i,1)=pop1_capraz_sayi(i,1)*1+pop1_capraz_sayi(i,2)*1/4+pop1_cap
raz_sayi(i,3)*1/16+pop1_capraz_sayi(i,4)*1/64;
populasyon_capraz_ondalik(1,i)=pop_capraz_ondalik(i,1);
populasyon_capraz_ondalik(2,i)=pop1_capraz_ondalik(i,1);
end
for i=1:n
f_capraz_dna(i)=tracklsq1(populasyon_capraz_ondalik(:,i));; % Fonksiyonun uygunluk
değeri hesaplanıyor.
end
uygunluk_capraz_deger=min(f_capraz_dna);

for i=1:n
ifuygunluk_capraz_deger==f_capraz_dna(i)
          k1_capraz=pop_capraz_ondalik(i,1);
          k2_capraz=pop1_capraz_ondalik(i,1);
end
end
    %Populasyona %30 Enzim ve virüs mutasyonu uygulanıyor.
w=randperm(n);
k=ceil(n*0.3); % Enzim ve virüs mutasyonu için DNA bazları seçiliyor.
for i=1:k
b=randseq(m);
pop_capraz(w(i),:)=b;
pop_mut=pop_capraz;
b=randseq(m+1);
        pop1_capraz(w(i),:)=b;
        pop1_mut=pop1_capraz;
       % pop1_mut=pop_mut
end
   %Yeni DNA dizileri sayıya çevriliyor. A:0,G:1,C:2,T:3
for i=1:n
for j=1:m+1
if pop1_mut(i,j)=='A'
           pop1_mut_sayi(i,j)=0;
elseif pop1_mut(i,j)=='G'
           pop1_mut_sayi(i,j)=1;
elseif pop1_mut(i,j)=='C'
```

```
            pop1_mut_sayi(i,j)=2;
elseif pop1_mut(i,j)=='T'
            pop1_mut_sayi(i,j)=3;
end
end
end
for i=1:n
for j=1:m
ifpop_mut(i,j)=='A'
pop_mut_sayi(i,j)=0;
elseifpop_mut(i,j)=='G'
pop_mut_sayi(i,j)=1;
elseifpop_mut(i,j)=='C'
pop_mut_sayi(i,j)=2;
elseifpop_mut(i,j)=='T'
pop_mut_sayi(i,j)=3;
end
end
end
     f1_mut_dna=zeros(n,1);
  % Diziler onluk sayıya çevriliyor.
pop_mut_ondalik=zeros(n,1);
   pop1_mut_ondalik=zeros(n,1);
for i=1:n
pop_mut_ondalik(i,1)=pop_mut_sayi(i,1)*16+pop_mut_sayi(i,2)*4+pop_mut_sayi(i,3)*1;
pop1_mut_ondalik(i,1)=pop1_mut_sayi(i,1)*1+pop1_mut_sayi(i,2)*1/4+pop1_mut_sayi(i,
3)*1/16+pop1_mut_sayi(i,4)*1/64;
populasyon_mut_ondalik(1,i)=pop_mut_ondalik(i,1);
populasyon_mut_ondalik(2,i)=pop1_mut_ondalik(i,1);
end
for i=1:n
f_mut_dna(i)=tracklsq1(populasyon_mut_ondalik(:,i)); % Fonksiyonun uygunluk değeri
hesaplanıyor.
end
uygunluk_mut_deger=min(f_mut_dna);
for i=1:n
ifuygunluk_mut_deger==f_mut_dna(i)
            k1_mut=pop_mut_ondalik(i,1);
            k2_mut=pop1_mut_ondalik(i,1);
end
end
ifuygunluk_deger<uygunluk_capraz_deger&&uygunluk_deger<uygunluk_mut_deger
Kp_dna=k1;
Ki_dna=k2;
elseifuygunluk_capraz_deger<uygunluk_deger&&uygunluk_capraz_deger<uygunluk_mut
_deger
Kp_dna=k1_capraz;
Ki_dna=k2_capraz;
else
```

```
Kp_dna=k1_mut;
Ki_dna=k2_mut;
end
if (uygunluk_deger<uygunluk_capraz_deger&&uygunluk_deger<uygunluk_mut_deger)
% Kp ve Ki değerleri belirleniyor.
f=f_dna;
kp=pop_ondalik;
ki=pop1_ondalik;
elseif
(uygunluk_capraz_deger<uygunluk_deger&&uygunluk_capraz_deger<uygunluk_mut_deg
er)
f=f_capraz_dna;
kp=pop_capraz_ondalik;
ki=pop1_capraz_ondalik;
else
f=f_mut_dna;
kp=pop_mut_ondalik;
ki=pop1_mut_ondalik;
end
end  % döngü sonlandı
Kp_dna
Ki_dna
```

ÖZGEÇMİŞ

Uğur ÇİĞDEM 1977 yılında Reşadiye'de doğdu. İlk, orta ve lise öğrenimini Reşadiye'de tamamladı. 1998 – 2002 yılları arasında Yakındoğu Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği (burslu) bölümünde okudu. 2002 yılında aynı bölümden bilgisayar mühendisi olarak mezun oldu. 2002 – 2006 yılları arasında çeşitli özel ve resmi kurumlarda çalıştıktan sonra 2006 yılında Gaziosmanpaşa Üniversitesi Reşadiye Meslek Yüksekokulunda öğretim görevlisi oldu. 2009 yılında Fırat Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği (yazılım) bölümünde yüksek lisans eğitimine başladı. Halen Reşadiye Meslek Yüksekokulunda öğretim görevlisi olup evli ve bir çocuk babasıdır.