T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK AMAÇLI GENETİK ALGORİTMA KULLANARAK BİYODİZİLERDEN ÇOKLU MOTİFLERİN KEŞFİ

Melikali GÜÇ

Tez Yöneticisi

Yrd. Doç. Dr. Mehmet KAYA

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ELAZIĞ, 2008

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK AMAÇLI GENETİK ALGORİTMA KULLANARAK BİYODİZİLERDEN ÇOKLU MOTİFLERİN KEŞFİ

Melikali GÜÇ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez, ... tarihinde aşağıda belirtilen jüri tarafından oybirliği / oyçokluğu ile başarılı / başarısız olarak değerlendirilmiştir.

Danışman: Yrd. Doç. Dr. Mehmet KAYA
Üye: Prof. Dr. Yakup DEMİR
Üye: Yrd. Doç. Dr. Ali KARCI

Bu tezin kabulü, Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ...../...../..... tarih ve ..... sayılı kararıyla onaylanmıştır.

TEŞEKKÜR

Bu tez çalışması boyunca bana yardımlarını esirgemeyen tez danışmanım Sayın Yrd. Doç. Dr. Mehmet KAYA'ya ve bu tez çalışması için proje desteği sağlayan FÜBAP'a teşekkürlerimi sunarım.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

SİMGELER VE KISALTMALAR LİSTESİ

A: Adenine bazı

YSA:Yapay Sinir Ağı

D: Sözlük

DNA: Deoksiribonükleik asit;

EM: Beklenen Değer Maksimizasyonu

G: Guanine bazı

GA: Genetik Algoritma;

I: Hizalama puanı

IC: Bilgi içeriği

i: Adım numarası;

k: Motifin test edilme sayısı

l: Kelime boyutu

m: Dizi boyutu

$m_1..m_n$: Motif numaraları

mRNA: Mesajcı RNA

N: Herhangi bir karakter

n: Sıra numarası;

p: Sözlük içerisindeki kelime çiftleri

RNA: ribonükleik asit;

rRNA: Ribosomal RNA

$S_1..S_n$:DNA dizileri.

T: Thymine bazı

TF: Transkripsiyon Faktörü

tRNA: Taşıyıcı RNA

u: motif uygunluğu

U: Uracil bazı

Σ: Alfabe

Γ: Dağılım dizisi

w:Motif uzunluğu

$w_1..w_n$:Motif ağırlıkları

∂ :Minimum benzerlik değeri

Ω :Ağırlık vektörü

η :Öğrenme oranı

U^* :Amaç

λ : Motifin bozulma miktarı

ε : Epsilon değeri

μ : Motif yoğunluğu

ÖZET

YÜKSEK LİSANS TEZİ

ÇOK AMAÇLI GENETİK ALGORİTMA KULLANARAK BİYODİZİLERDEN ÇOKLU MOTİFLERİN KEŞFİ

Melikali GÜÇ

Fırat Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

2008, Sayfa: 60

Bu tezde DNA üzerinde bulunan saklı bilgilerin, genetik algoritma yardımıyla çıkartılması gerçekleştirilmiştir. DNA içerisinde gizli örüntü çıkarma genetik bilimi açısından çok önemli bir konudur. Biyolojik diziler üzerinde arama yapan birçok yöntem bulunmaktadır. Bu yöntemlerin birçoğu tek bir amaca ulaşmak için çalışmaktadır. Bazı problemlerde birden fazla amacı gerçekleştirmek gerekir. Bu gibi durumlarda tek amacı optimize eden yöntemler istenilen sonucu vermez.

Bu tezde önerilen yöntem DNA üzerinde motif araması yaparken birden fazla amacı gerçekleştirir. Yöntem daha önce geliştirilmiş yöntemlerden bu açıdan farklılık gösterir. Geliştirilen bu algoritma birçok giriş parametresine bağlı en iyi çözüm kümesini sunar. Çözüme etki eden değişken değerlerini kendisi otomatik olarak ayarlar.

Geliştirilen bu yöntem ile DNA üzerindeki çok iyi gizlenmiş motifleri arama işleminde daha optimal sonuçlar elde edilecektir.

Anahtar Kelimeler: Kümeleme yöntemleri, motif keşfetme, çok amaçlı genetik algoritma, biyoenformatik.

ABSTRACT

MS THESIS

DISCOVERING MULTIPLE MOTIFS FROM BIOSEQUENCES USING BY MULTI-OBJECTIVE GENETIC ALGORITHM

Melikali GÜÇ

Fırat University

Graduate School of Natural and Applied Sciences

Department of Computer Engineering

2008, Page: 60

In this thesis, information that is hidden on DNA was carried out by helping of genetic algorithm. Bringing out the hidden pattern is important for genetics. There are a lot of methods that is searching on biological concordance. Most of these methods study to get only one objective. It is a necessity to get more than one objective on some problems. In such conditions, the methods don't work out that optimize only one objective.

The method that is recommended on this thesis carries out more than one objective when it is searching motif on DNA. So, this method is different from the others that were developed before. This developed algorithm offers the best solutions that depended on multiple input parameters. It arranges the amount of the affecting factors automatically. We get more optimal solutions with this developed method on searching very good hidden motifs on DNA.

Keywords: Clustering techniques, motif discovery, multi-objective genetic algorithm and bioinformatics.

1. GİRİŞ

DNA (Deoksiribonükleik Asit) tüm canlı hücrelerinde hücresel işlemleri yöneten nükleik asit dizisidir. DNA, hücre içerisinde protein sentezini gerçekleştirirken aynı zamanda canlıların özelliklerinin nesilden nesile aktarılmasını da sağlar.

DNA, canlıya ait tüm bilgilerin depolandığı veri bankasıdır [1]. Canlının gelişmesiyle ilgili tüm bilgiler gizlenerek kodlanmıştır. Genetik bilimin gelişmesiyle DNA haritası çıkarılmıştır [2]. Yine de DNA'nın içindeki gizli bilgiler tam olarak bilinmemektedir. DNA'nın işlevinin keşfedilmesi, birçok hastalığın da tedavisinin bulunmasını sağlayacaktır. Kalıtsal olarak aktarılan hastalık ve özürler önceden tespit edileceğinden, birey dünyaya gelmeden bozuk alan düzeltilmeye çalışılacaktır [3]. DNA'nın yapısının çözülmesi ile protein sentezi daha kolay yorumlanabilecektir.

DNA üzerinde araştırma yapmak, uzun ve maliyetli bir iştir. Bir insan DNA'sında 300 milyar baz çifti olduğu varsayılırsa, bu uzunluktaki bir dizide gen analizi yapmak insan yeteneklerinin sınırlarını aşmaktadır. Dolayısıyla çok kısa sürede çok fazla işlem yapan bilgisayar sistemleri, analiz aşamasında vazgeçilmez olmuştur.

Bilişim dünyasındaki gelişmeler ile daha karmaşık ve hızlı bilgisayar sistemleri gelişmiştir. Bu alandaki ilerleme, zaman içerisinde genetik bilimindeki ilerlemeye de destek sağlamıştır. Bu iki alanın ortak bir çalışma sahası olan biyoenformatik, bilim dünyasında uzun zaman önce yerini aldı. Biyoenformatik, hem DNA yapısıyla hem de protein yapıları üzerinde çalışma yapmaktadır.

Biyoenformatik alanında çalışan araştırmacılar, zamanla gelişen analiz tekniklerini kullanmaktadır. Bilgisayar bilimlerinde önerilen yeni metotlar, biyoenformatik alanında da kullanılmaktadır[4-7]. Burada araştırmacılar analiz için güçlü yöntemleri tercih etmektedir. Dolayısıyla iyi bir analiz ve çıkarım yöntemi ile DNA üzerinde araştırma yapmak, elle tutulur sonuçlar doğuracaktır. Biyoenformatikte meydana gelen ilerlemeler ile birlikte çok sayıda gen bankası kurulmuştur [8-10]. Burada toplanan bilgiler insanların ve uygulama geliştiricilerin kullanımına sunulmuştur. Bu alanda geliştirilen birçok tekniğin uygulaması geliştirilerek uzaktan hizmet vermektedir. Kullanıcılar verilerini internet üzerinden sunuculara gönderip sonuçları yine internet üzerinden alabilmektedir. Bu sayede büyük bir bilgi ağı da oluşturulmuştur. Bu alandaki yöntemlerin birçoğu genelde birbirinden etkilenerek geliştirilmiştir.

Bu tezde DNA üzerinde araştırma yapan bir yöntem çalışılmıştır. Yöntem geliştirilirken, bu alanda öne sürülen diğer yöntem ve teknikler dikkatli bir şekilde takip edilmiştir.

1.1. Tezin Amacı

Bu tezde DNA üzerinde gen analizi yapan bir yöntem geliştirilmiştir. Önerilen yöntem DNA dizileri üzerinde çoklu motifleri keşfetmeyi hedeflemektedir. Biyoenformatik alanında öne sürülen birçok yöntem, tek bir amacı optimize etmeye çalışırken, bu tezde önerilen yöntem birden fazla hedefe ulaşmaya çalışmaktadır [11-13]. Yöntem birden fazla DNA dizisini giriş verisi olarak alır ve sonuçta gizli veya iyi saklanmış motif örüntülerini çıktı olarak verir. Aşağıdaki alt bölümlerde motif keşfi için öncelikle biyoenformatiğin temel kavramları üzerinde durulacaktır.

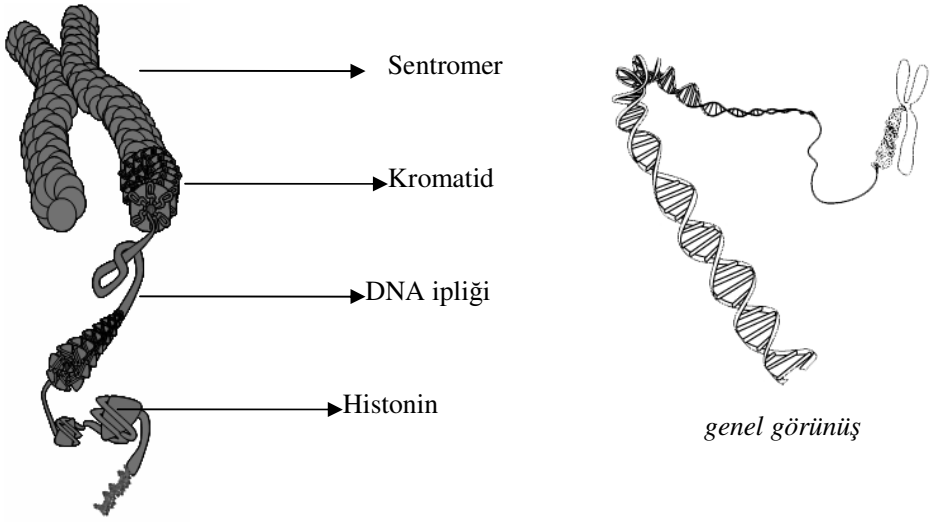
1.2. Genetik Bilimindeki Temel Kavramlar

Canlıların var olan özelliklerini, kendinden sonra gelen nesillere aktarmasını inceleyen bilim dalına "*genetik"* denir. 1990 sonrası genetik alanında ciddi çalışmalar başlamıştır. Bu yılda "İnsan Genom Projesi" hayata geçirilmiştir [14]. Proje kapsamında ilk olarak DNA haritası çıkartılması hedeflenmiştir. DNA haritasının çıkarılması ile bu alanda büyük bir adım atılmıştır. Projenin ilerleyen aşamalarında DNA yapısının ve görevinin tam olarak çözülmesi hedeflenmektedir. Bu işlem gerçekleştiğinde artık hastalıklar daha önceden, kişi doğmadan öğrenilebilecek ve ona göre bir çözüm üretilecektir. Ayrıca ölümcül hastalıklara neden olan virüslerin yapısı da kolay bir şekilde çözülüp virüs yok edilecektir [15]. Yukarıda bahsedilen nedenlerden dolayı DNA'nın şifrelerinin çözülmesi genetik bilimi ve insanlık açısından büyük önem taşımaktadır.

1.3. DNA'nın Yapısı, Görevi ve Özellikleri

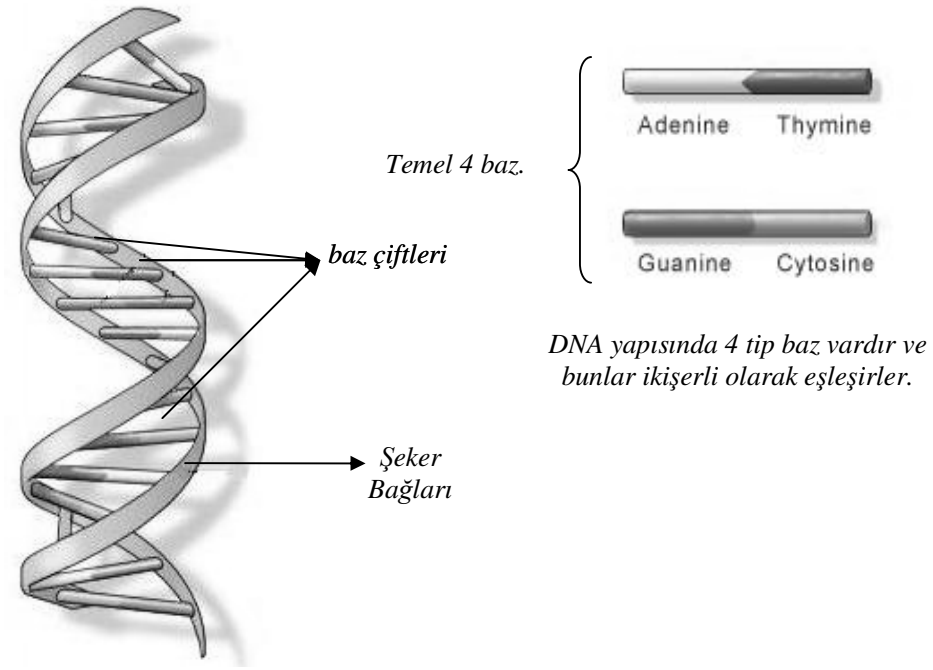
Canlı hücrelerinin tümü, DNA adı verilen nükleik baz dizisine sahiptir. DNA tek hücreli canlılarda ve virüslerde hücre sitoplâzmasında, çok hücreli canlılarda (insan, bitki gibi) ise hücre çekirdeğinde bulunur. DNA insana ait göz rengi, boy uzunluğu, ses kalınlığı, ten rengi, saç rengi, hayatı boyunca yakalanacağı hastalıklar veya hangi hastalıklara karşı zayıf olacağı gibi birçok bilgiyi şifrelenmiş bir şekilde kodlar [16].

DNA sarmalının Histon adındaki baz etrafına kıvrılarak oluşturduğu tekil yapıya *kromozom* denir. İnsan DNA'sında 46 kromozom bulunur. Köpekte 78, kurbağada 26 farede ise 42 kromozom bulunur. Hücre bölünmesi aşamasında, anne ve babadan 23 kromozom alınır ve bunlar eşleştirilerek tekrar 46 kromozom üretilir. Kromozom, uzunluğu 25 ile 30 bin arasında değişen gen parçalarıdır. Şekil 1.1'de kromozomun temel yapısı gösterilmiştir.



Şekil 1.1: Kromozom yapısı.

DNA yapısında 4 tip baz bulunur. Bunlar Thymine, Adenine, Guanine, Cytosine olarak isimlendirilir. Bu 4 baz merdiven şeklindeki bir yapıda karşı karşıya gelecek şekilde birleşerek dururlar. Thymine her zaman Adenine ile Guanine ise her zaman Cytosine ile birleşir. Şekil 1.2'de DNA yapısındaki bazlar gösterilmiştir.

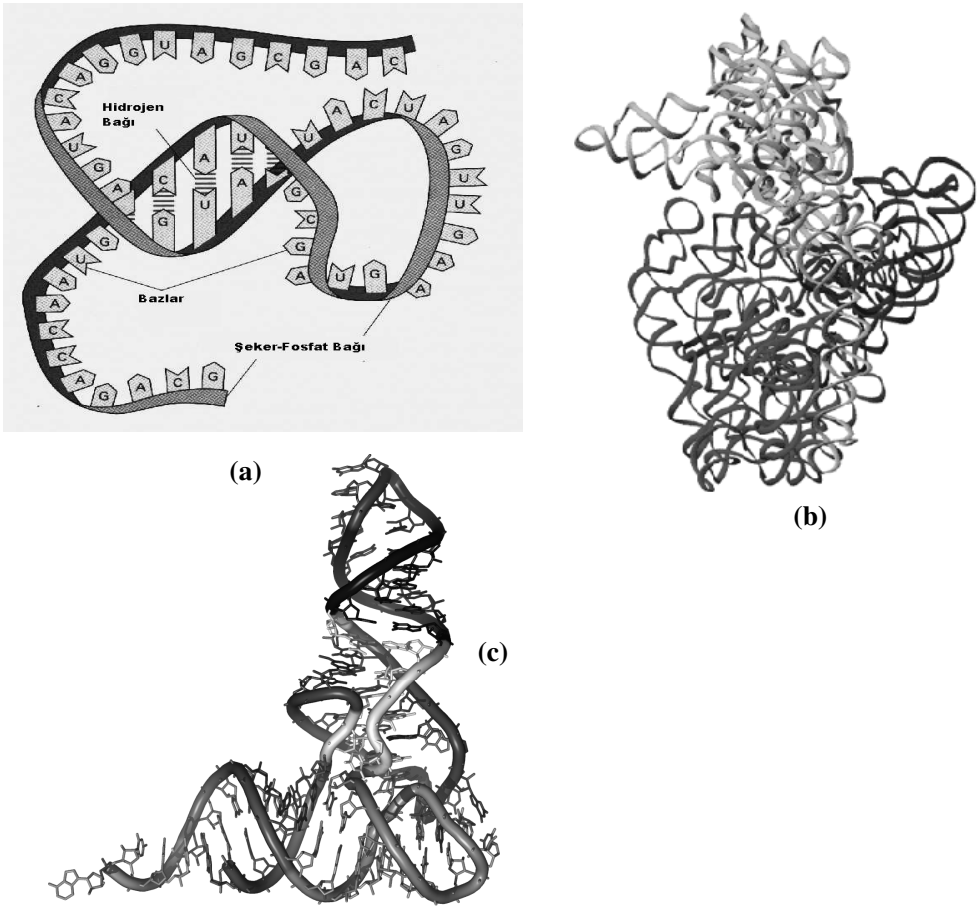


Şekil 1.2: DNA yapısındaki bazlar.

1.4. RNA

RNA(Ribonükleik Asit) DNA'nın protein sentezi yapmak için kullandığı tek zincirli nükleotid dizisidir. 3 tip RNA vardır; Bunlar tRNA, mRNA ve rRNA'dır [17]. tRNA, hücre sitoplâzmasında serbest halde dolaşan aminoasitleri yakalar ve ribozoma getirir. Ribozom yakalanan bu amino asitleri protein üretimi için kullanır. mRNA, DNA'dan gelen bilgiyi ribozoma getirmek için kullanılır. DNA hücre dışına çıkmaz. Bunun yerine dışarıya mRNA aracığıyla mesaj gönderir. DNA'nın kopyalanması işlemine "DNA ikileşmesi" denir [18]. rRNA ribozom organelinin büyük oranda yapısını meydana getiren RNA türüdür.

RNA bazı yönlerden DNA ile farklılık gösterir: RNA tek zincirli DNA ise çift zincirlidir. RNA'da riboz bulunur, DNA'da deoksiriboz bulunur. DNA'da Timin, RNA'da bunun yerine Urasil vardır. DNA yapısal olarak kendi başına değişmez veya silinmez. RNA protein üretim sürecinden sonra dağılır. Şekil1.3'de değişik yapıda RNA'lar görülmektedir.



Şekil 1.3: RNA'nın değişik yapıları. **(a)** RNA'nın genel yapısı (mRNA), **(b)** rRNA, **(c)** tRNA.

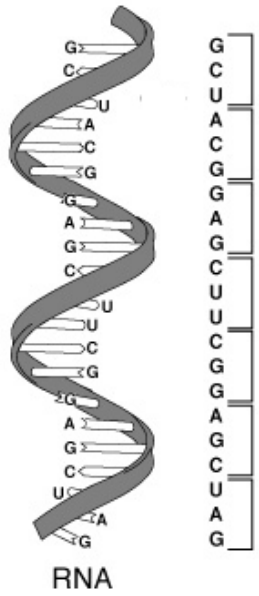
1.5. Proteinlerin Yapısı ve Görevi

Bir dizi aminoasidin birleşmesiyle oluşan yapıya "protein" denir. Proteinler hücre içerisinde değişik görevler üstlenmişlerdir. Hücresel faaliyetlerin tetiklenmesinde görev alırlar. Bazı proteinler enzim olarak görev yaparlar. Bu durumda katalizör davranışı gösterir ve kimyasal tepkimeleri hızlandırır veya yavaşlatırlar. Bazı proteinler hücre içi kimyasal tepkimelerde aktif rol alırlar. Örneğin hormon salgılarlar veya hormon seviyesini kontrol ederler. Hücreler arası madde taşınması bir başka görevidir. Vücuda giren mikroplara karşı savunma yapmak bir başka görevidir. Hücrelerin bölünmesinde aktif rol alırlar. Yani hücrelerin yapı taşıdırlar [19].

DNA'dan gelen protein bilgisi mesajcı RNA(*mRNA*) üzerine kodonlar ile kodlanmıştır. Bir kodon 3 adet bazın yan yana getirilmesiyle oluşur. Dolayısıyla 4^3=64 farklı kodon vardır. Ribozom, mRNA'yı okurken 3'erli gruplar halinde okur. Her kodon bir aminoaside karşılık gelir. Şekil 1.4'de kodon tablosu verilmiştir.

		2.BAZ					
		U	**C**	**A**	**G**		
1.BAZ	**U**	UUU Phe	UCU Ser	UAU Tyr	UGU Cys	**U**	**3.BAZ**
		UUC Phe	UCC Ser	UAC Try	UGC Cys	**C**	
		UUA Leu	UCA Ser	**UAA Bit.**	**UGA Bit.**	**A**	
		UUG Leu	UCG Ser	**UAG Bit.**	UGG Trp	**G**	
	C	CUU Leu	CCU Pro	CAU His	CGU Arg	**U**	
		CUC Leu	CCC Pro	CAC His	CGC Arg	**C**	
		CUA Leu	CCA Pro	CAA Gln	CGA Arg	**A**	
		CUG Leu	CCG Pro	CAG Gln	CGG Arg	**G**	
	A	AUU Ile	ACU Thr	AAU Asn	AGU Ser	**U**	
		AUC Ile	ACC Thr	AAC Asn	AGC Ser	**C**	
		AUA Ile	ACA Thr	AAA Lys	AGA Arg	**A**	
		AUG Met	ACG Thr	AAG Lys	AGG Arg	**G**	
	G	GUU Val	GCU Ala	GAU Asp	GGU Gly	**U**	
		GUC Val	GCC Ala	GAC Asp	GGC Gly	**C**	
		GUA Val	GCA Ala	GAA Glu	GGA Gly	**A**	
		GUG Val	GCG Ala	GAG Glu	GGG Gly	**G**	

(a)



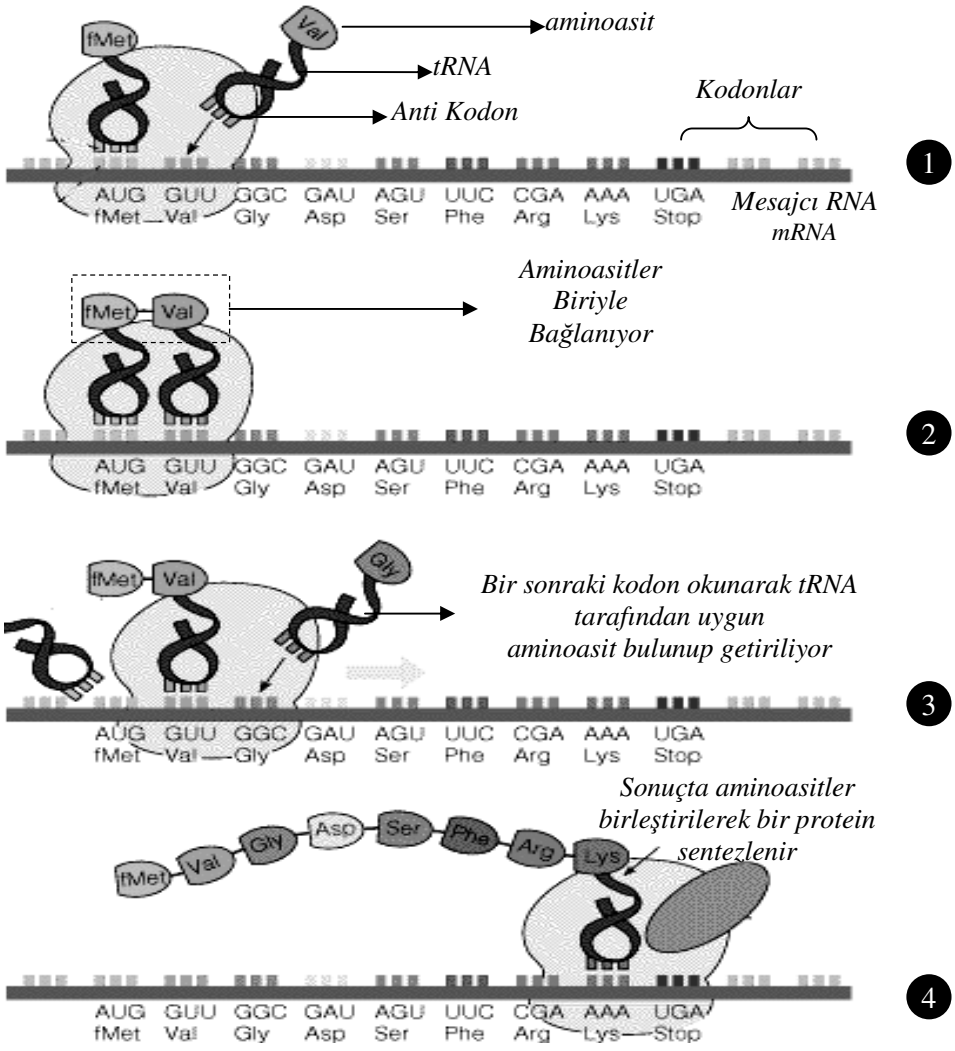
(b)

Şekil 1.4: (a) Kodon kodlama tablosu **(b)** mRNA'nın kodonlara ayrılması.

Toplam 64 kodon olmasına rağmen bunların 3 tanesi başlama ve bitiş kodonu olarak çalışır. İnsan hücrelerinde 497 tane tRNA protein sentezinde görev alır. Buna karşılık 48 amino asit protein yapımında kullanılır. mRNA'daki kodona karşı tRNA'da antikodon bulunur.

1.6. Protein Sentezi

Protein sentezi, DNA sarmalının açılıp bir bölgesinin kopyalanması ile başlar. Kopyalanan bu baz dizisine mRNA denir. Protein sentezi ribozom içerisinde gerçekleşir. DNA'dan mRNA ile gelen komut ribozom içerisinden geçer. Bu geçiş sırasında mRNA 3'erli gruplar (kodonlar) halinde okunur. Kodona karşılık gelen amino asidi tRNA sitoplâzmadan bulup getirir. Bu aminoasidi protein zincirine bağlar. Daha sonra bir sonraki kodon okunur ve tRNA buna karşılık gelen amino asidi sitoplâzmadan bulur ve biraz önce eklenen aminoaside bağlar. Bu şekilde durdurma kodonu bulunana kadar mRNA sürekli kodonlar halinde okunur [20]. Şekil 1.5'de protein sentezi gösterilmiştir.



Şekil 1.5: Protein sentezi.

Protein sentezi biyoenformatik açısından çok önemlidir. Çünkü tüm hücresel olaylar protein sentezi ile yapılmaktadır. Bazen DNA yapısı dış etkenler tarafında bozulabilir veya kalıtım yoluyla bazı genler hatalı olarak aktarılmış olabilir. Bu durumda DNA hatalı protein sentezleri yapar. Bu da hücrenin görevi dışında başka bir görev yapmasına neden olur. Örneğin kanser olmuş bir karaciğer hücresi, şeker depolaması gerekirken sürekli bölünmeye çalışır. Bu da hastalıklı kişinin ilerleyen zaman diliminde ölümüne yol açar. Eğer bu hatalı bölünmeye neden olan gen parçası ve protein yapısı iyi analiz edilirse, ilerleyen yıllarda gelişen teknoloji ile birlikte DNA içerisindeki bu yıkıcı etkiye sahip alan düzeltilebilir veya tamamıyla silinebilir. Bu da hastanın yaşama şansını yükseltir veya ölüm riskini bütünüyle kaldırabilir. Dolayısıyla DNA'nın işlevini öğrenmek için protein sentezi iyi incelenmelidir.

1.7. Teze Bakış

Bu tezin ilk bölümünde genetik bilimi ve genetik ile bilişim teknolojileri arasındaki güçlü bağlantı vurgulandı, genetik ile ilgili temel kavramlar hakkında kısaca bilgi verildi. İkinci bölümde, DNA üzerinde motif keşfetmek için kullanılan temel algoritmalar sınıflanarak anlatılacaktır. Üçüncü bölümde, motif keşfinde evrimsel algoritmaların nasıl kullanıldığı hakkında bilgi verilecektir. Dördüncü bölümde, bu çalışmada geliştirilen çok-amaçlı genetik algoritma anlatılacaktır. Beşinci bölümde, algoritmanın kullanıldığı uygulama hakkında bilgi verilecek ve uygulamanın çalıştırılması ile elde edilen sonuçlar değerlendirilecektir. Altıncı bölüm sonuç bölümüdür, burada algoritmanın genel değerlendirilmesi yapılacaktır.

2. MOTİF ÇIKARMA ALGORİTMALARI

DNA veya RNA dizileri içerisindeki gizli gen örüntülerine *motif* denir [21]. Farklı dizilerde bulanan gen parçaları aynı protein sentezini gerçekleştirebilirler. Bir gen içerisinde birden fazla motif bulunabilir. Motif uzunluğu sentezlenen proteinin boyutuna göre değişiklik gösterebilir.

Motifler farklı diziler içerisinde farklı pozisyonlarda bulunabilirler. Çoklu dizi sıralama algoritmaları bu motif pozisyonlarını bulmak için dizileri sağa veya sola ötelerler [22, 23]. Motif içerisindeki baz dizileri birbiriyle tam benzerlik göstermeyebilir. Bu durumda birçok yaklaşım hizalamayı belli bir miktar hata ile kabul eder [24, 25].

Motiflerin geçerliliğini veya tutarlılığını bulmak için ağırlık matrisinden faydalanılır. Bu matris içerisinde bazların sütunlardaki frekansları tutulur. Daha sonra bu matris kullanılarak motif için hizalama puanları hesaplanır. Tüm sütunlardaki puanlar toplanarak motifin toplam benzerlik puanı bulunur. Eğer motif benzerliği eşik değerinin altında ise motif başarısız sayılır ve tekrardan motif araması yapılır. Şekil 2.1'de diziler içerisinde bulunan bazı motif örnekleri verilmiştir.

Yapılan araştırmalar sonucunda bazı motiflerin protein sentezini başlattığı görülmüştür. Bu motifler "transkripsiyon faktörleri" olarak adlandırılır. Bunlar DNA üzerindeki kodlanmış bilgiyi okuyarak buna göre çıkarım yaparlar [26]. Gen kopyalanmasını ya arttırırlar yada azaltırlar. Bundan dolayı hücresel işlemlerin gerçekleşmesinde önemli rol alırlar. Başlıca transkripsiyon faktörleri şunlardır: TATA box [TATAAA], BRE [(G/C) (G/C) (G/A)CGCC], Inr [TCA(G/T) T(T/C)], ve DPE [(A/G) G(A/T) (C/T) (G/A/C)] olarak bilinir.

Dizi 1:	T	A	G	T	T	A	A	G	G	C	T	C	T	T	G	A	G	G	A	T
Dizi 2:	C	C	A	A	T	T	A	A	G	A	A	G	G	C	T	A	A	G	C	T
Dizi 3:	G	G	A	C	A	T	G	G	C	T	T	G	C	G	G	A	A	C	C	T
Dizi 4:	A	A	T	C	C	T	A	A	G	A	T	A	T	G	A	A	T	G	C	T

Olası Motif Örneği:

Parça 1	A	A	G	G	C	T
Parça 2	A	A	G	G	C	T
Parça 3	A	T	G	G	C	T
Parça 4	A	A	T	C	C	T
Parça 5	A	A	T	G	C	T
Uzlaşı Motifi	A	A	G	G	C	T

Bir dizide birden fazla motif bulunabilir. Bunun yanı sıra aynı motif örneği birden fazla dizi içerisinde de bulunabilir.

Şekil 2.1: Dizi içerisindeki olası motif örnekleri.

2.1. Deterministik Yaklaşımlı Algoritmalar

2.1.1. YMF, Oligoanalysis:Örüntü-Güdümlü Dizge Sayma

En basit motif çıkarma algoritmalarından birisi olan bu yaklaşımda, dizi içerisinde boyutu *l* olan tüm kelimelerin bir listesi oluşturulur. Motif uzunluğu, *l* ile dizi boyutu arasında değişir. Bu yaklaşımın temelinde, her bir kelimenin dizi başından sonuna kadar birer kaydırmak suretiyle karşılaştırılması yatar [27]. Bu işlem listedeki tüm kelimeler için tekrarlanır. Herhangi bir dizgenin bulunmasında bir veya daha fazla uyumsuzluk çıkması olasıdır.

Bu algoritma hesaplama yapmak için l^4 byte hafıza alanı ve hızlı bir donanım mimarisine ihtiyaç duyar. Algoritma ile işlem yapmak motif keşfinin ilk öne sürüldüğü 1985 yılında oldukça güçtü [28]. O dönemde bilgisayar sistemlerinin alt yapısı ile hesaplama, uzun ve maliyetli bir işti. Günümüz bilişim dünyasında bu durum değişti. Bilgisayar sistemlerinin gelişmesiyle birlikte hafıza boyutları ve işlemci hızları arttı. Bununla birlikte işletim sistemlerinin hafıza kullanma yöntemleri de algoritmanın performansını etkileyen faktörlerden birisidir.

Motif tarayıcılar tarafından bulunan motiflerin uzunluğu, sıklıkla 6 bazlı nükleotidlerdir. Algoritma, 6 uzunluklu 64=1296 farklı motif için, dizi içerisinde hesaplama yapar. Daha gelişmiş algoritmalarda bu uzunluk artabilir.

Algoritmanın çalışma prensibi:

Adım 1: *l* uzunluk olabilecek tüm baz dizilerinin(motiflerin) listesini oluştur.

Adım 2: Listenin başındaki ilk elamanı bul.

Adım 3: Motifi dizi başından sonuna kadar birer kaydırarak test et.
Test aşamasında motifin uygunluğunu hesapla.

Adım 4: Listedeki bir sonraki elamanı bul. Eğer listedeki motiflerin hepsi test edilmişse Adım 5'e, test edilmemişse Adım 3'e git.

Adım 5: Sonuçları göster.

Algoritma motif uzunluğu *l* ve dizi uzunluğu *m* için; $l^4 *(m+ l- 1)* l$ adet hesaplama yapar. Örneğin 100 uzunluklu bir dizi ve motif uzunluğu 6 için algoritma karmaşıklığı:

Bir motifin test sayısı→ (100+6–1)*6=630,

Listedeki birey sayısı→ 6^4=1296,

Toplam test sayısı→1296*630=816480 olarak bulunur.

Şekil 2.2'de algoritma adımları gösterilmiştir.

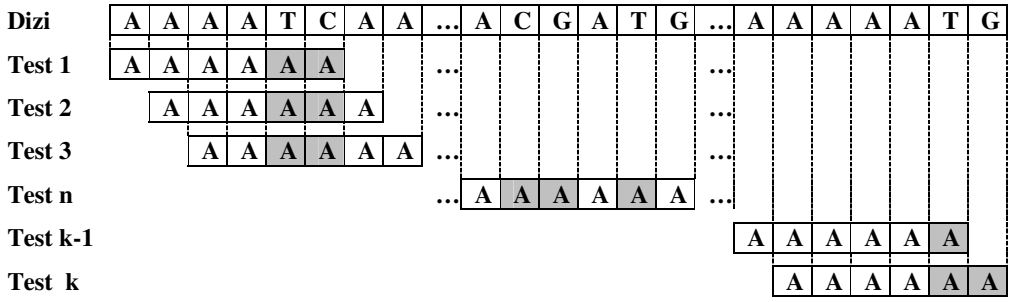
Adım 1: 6 Uzunluklu baz dizilerinin(motiflerin) listesi oluşturulur:

AAAAA	AAAAAT	AAAAAG	.	.	.	.	ACCCCG	ACCCCC
TAAAAA	TAAAAT	TAAAAG	.	.	.	.	TCCCCG	TCCCCC
GAAAAA	GAAAAT	GAAAAG	.	.	.	.	GCCCCG	GCCCCC
CAAAAA	CAAAAT	CAAAAG	.	.	.	.	CCCCCG	CCCCCC

Adım 2: İlk birey bulunur:

A	**A**	**A**	**A**	**A**	**A**

Adım 3: Motifin Testi:



Motif uygunluğunun hesaplanması:

Test No	Test 1	Test 2	Test 3	.	.	Test n	.	.	Test k-1	Test k
Uygunluk	4/6	4/6	4/6	.	.	3/6	.	.	5/6	4/6

Ortalama Uygunluk= $\frac{\sum_{i=1}^{k} u_i}{k}$ ile hesaplanır. Burada ***u*** motifinin ***i.*** adımdaki uygunluğu,

k motifin test sayısıdır.

Adım 4: Bir sonraki elamanı bul ve liste sonuna gelinip gelinmediğini test et:

Sonraki motif→ **A A A A A T**

Liste sonuna gelinmedi Adım 3'e git.

Adım 5: Sonuçları göster:

Motif						Ortalama Uygunluk
A	**A**	**A**	**A**	**A**	**A**	**4/6**
A	**A**	**A**	**A**	**A**	**T**	**4/6**
A	**A**	**A**	**A**	**A**	**G**	**2/6**
.	**.**	**.**	**.**	**.**	**.**	
.	**.**	**.**	**.**	**.**	**.**	
G	**G**	**G**	**T**	**A**	**C**	**1/6**
.	**.**	**.**	**.**	**.**	**.**	
C	**C**	**C**	**C**	**C**	**A**	**1/6**
C	**C**	**C**	**C**	**C**	**T**	**2/6**
C	**C**	**C**	**C**	**C**	**G**	**2/6**
C	**C**	**C**	**C**	**C**	**C**	**2/6**

Şekil 2.2: Uzunluğu m olan bir dizi için algoritma adımları.

2.1.2. Örnek Güdümlü Sayma

Örnek veya dizi güdümlü yöntemler benzer özellikler sergiler [29, 30]. Her iki yaklaşımda da, mümkün olabilecek tüm örneklerin listesi oluşturulmaz. Bunun yerine işlem yapılan diziden belli aralıklarla belli uzunlukta parçalar alınır ve alınan parçalarının bir dizisi oluşturulur. Oluşturulan motif listesindeki her bir birey sırayla algoritma adımlarından geçirilerek sonuçlar elde edilir.

Bu yaklaşımda her bireyin en az bir olumlu sonucu vardır. Bunun yanı sıra dizi içerisinde bulunma olasılığı da artmış olur. Bir gen dizisinin belli bir görevi vardır, dolayısıyla dizi içerisindeki motifler benzer baz çiftlerine sahiptirler. Bu algoritmadaki eksiklik çok iyi gizlenmiş örüntülerin bulunamama olasılığıdır. Zayıf ürünlerin dizi içerisinde bulunup güçlü ürünlerden bir tanesinin bulunmaması örnek olabilir.

Moleküler Yörünge Paketi (MOPAC) algoritması başka bir puanlama gereksinimi duymadan sıra dışı bir filtreleme yöntemi kullanarak pozitif dizilerin l uzunluktaki alt dizgelerini numaralandırır ve negatif dizi içerisinde verilen tüm l-alt dizgeyi silerek az sayıda sonuç tutar. Sonra verilen dizgeler ile diğer tüm dizgeler arasındaki mesafeyi hesaplar. Bir sonraki adımda tek bir aykırı değerin algoritma karmaşıklığını artırma varsayımı temeline dayanan mesafe ölçüm işleminde eğer mesafe büyük ise dizge silinir. MOPAC algoritması, sonunda diziyi birbirine benzeyen dizge kümelerine böler. Dizgeler belli bir eşik değerine ulaşıncaya kadar örüntüler arasındaki *hamming* mesafesine göre kümelenirler. Her bir gruptan son bir karar verilir ve kullanıcıya gösterilir.

Örnek ve örüntü güdümlü yollar ortak kelime araması yapmak için temel yöntemlerdir. Bu yaklaşımlar, eksiksiz bir numaralandırmaya izin veren hesaplama gücünden dolayı eskidirler.

2.1.3. Teiresias: Dizge Katlama Yaklaşımı

Bu algoritma tüm diziler içerisinde yer alan kısa örüntülerin listesiyle başlar. Bir çiftleştirme adımıyla birleştirilir ve herhangi bir harf anlamındaki c özel karakterini içerir. Daha sonra algoritma daha uzun örüntüleri bulmak için katlama aşamasında bunları birbiriyle yapıştırır. Genişleme işlemi, daha özel olanların ilk önce, bir veya daha fazla özel sembole sahip olanların sonda bulunduğu basit örüntü sıralaması ve birinin ön eki diğerinin son eki olan örüntülerin aranması ile yapılır [31].

Tasarım açısından algoritma, her motif örneğinin her dizide bulunduğunu varsayar. Bu algoritma sözlük temelli yaklaşım ile benzerlik gösterir.

2.1.4. Moby Dick: Sözlük Temelli Yaklaşım

Bu yöntem kelimelerden meydana gelen bir *D* sözlüğü ile başlar. Daha sonra *D* sözlüğünde düşük *P* değerlerine sahip sıralanmış *p* çiftlerini arar ve *p*'yi *D*'ye ekleyerek *D*'yi günceller. Eğer *D* içerisinde düşük *P* değeri varsa tüm p çiftlerini test eder. Başlangıç sözlüğü A,C,T,G harflerini ve bunların *P* değeri olarak bilinen frekanslarını içeren basit yapıdadır. Burada amaç, sıra dışı genom dizisini kullanarak sıra dışı örüntüleri bulmaktır [32].

Bu algoritma araştırma yapılan motif uzunluğunu optimize eder. Bir başka deyişle sabit uzunlukta motif aramaktansa motif sayısını değişken tutup motifin bulunma olasılığını artırır.

2.1.5. Consensus: Profil Numaralandırma

Uzlaşı olasılık matris modeli yazarları tarafından geliştirilmiştir [33]. Bu yöntem, matrissel ve deterministik yaklaşımların bir karışımıdır. Birçok arabirim tarafından iyi bilinir ve desteklenir. İki türü mevcuttur; bunlardan bir tanesi sabit uzunluklu motifleri bulmak için arama yapar. Diğeri ise optimal uzunluğa kendisi karar verir yalnız bu yapılırken fazladan parametreye ihtiyaç duyar.

Daha önceden tanımlanmış *w* uzunluğundaki bir motifin aranması aşağıda belirtilen adımlar gerçekleştirilerek yapılır:

w uzunluğundaki bir *s* alt-dizgeyi için, bu dizgeyi yansıtan bir matris hesaplanır. Daha sonra bu matris kullanılarak, algoritma diğer benzer dizge kümeleri için dizi tekrar taranır. Bu kümeden orijinal *s* dizgeyi ile birlikte bir matris oluşturulur. Farklı başlangıç noktaları kullanılarak çok faklı matrisler elde edilebilir. Algoritma maksimum "Bilgi içerikli" matrisleri tutar. Bu liste kullanılarak algoritma adımları tekrarlanır, filtrelenen dizgeler yeni matrislere sebep olan matrislere durdurma koşulu sağlanana kadar eklenmeye devam eder. Durdurma şartları, matris için alan sayısı veya tüm dizi katılımlarının alan sayısı limiti olabilir. Sonuç olarak algoritma, matrislerin bilgi içeriklerini matrislere alan ekleyerek geliştirmeye çalışır.

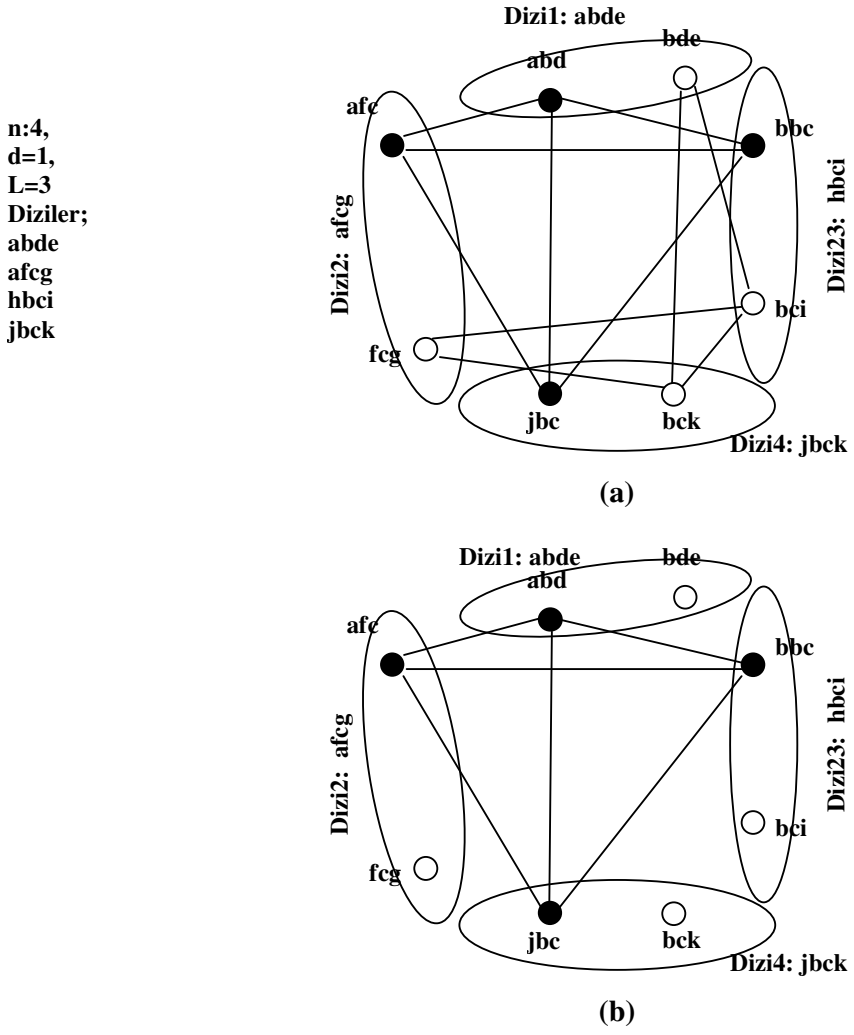
Eğer kullanıcı tarafından motif uzunluğu belirtilmemiş ise; kullanıcı fazladan parametre gerektiren bir uygunluk yerine özel bir uygunluk kullanmalıdır. Çünkü motif uzunluğuna bağlı olarak algoritma bilgi içeriğini direkt olarak lineer bir şekilde maksimum yapamaz.

2.1.6. Winnower, SP-STAR, cWinnower: Klik-Tabanlı Yaklaşımlar

Çizge tabanlı bir yaklaşım olan bu algoritma, tüm alt dizgelerin listesini oluşturur. Alt dizgeler çizge içerisinde köşe olarak gösterilir. Köşeler tüm dizilere bağlı olarak gruplara ayrılır.

Eğer herhangi iki köşe diğer parçalara göre benzerlik gösteriyor ise bu iki köşe kenar yardımıyla birbirine bağlanır [34]. Burada benzerlik belirli bir uzunluktan daha iyi olan "*Hamming Mesafesi*" anlamına gelir.

Algoritma daha sonra çizge içerisindeki klikleri bulmaya çalışır. Bir klik tamamen birleştirilmiş düğümlerden oluşur. Şekil 2.3 incelendiğinde; eğer **afc abd** ile , **abd jbc** ile, **jbc** ise **afc** ile benzerlik gösteriyorsa **afc abd** ve **fbc** ayrıtları da vardır. Bunlar kapalı üçgen şekline benzeyen 3lü klik formundadır. Son olarak algoritma *maksimum klik* parçası olmayan tüm ayrıtları siler. Ana fikir k-kliğin bir parçası olmak, aslında bir ayrıtın en az belirli bir sayıda genişleyebilir kliğin parçası olmasıdır. Genişleyebilir klik azami klikten daha küçüktür. Bu nedenle bu kritere uymayan tüm kenarlar silinir. Algoritma bu sebeple *k*'nın düşük değerleriyle başlar ve tekrarlayarak işlemi sürdürür.

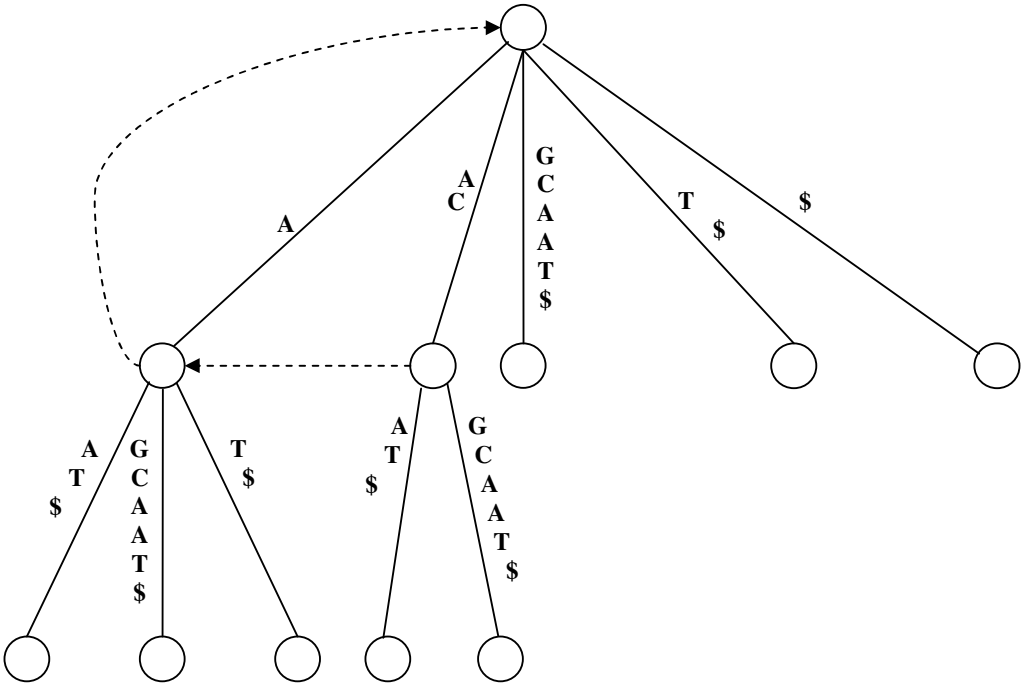


Şekil 2.3: Winower kliksiz düğümlerden kenarları siler.

2.1.7. SMILE: Sonek Ağaçları

Bir sonek ağacı, bir metnin kelimelerine hızlı bir şekilde ulaşmak için kullanılan bir özel indeks, veri yapısıdır. Bu ağaçlar tekrar eden kelimeleri bulabilir. Bu yüzden motif keşfi için ideal bir yöntemdir. Ağaç kendi başına çözüm uzayı oluşturamaz, yalnızca dizi-dizgeler için hızlı erişim sağlarlar. Önek ağaçları hafıza alanında inşa edildikten sonra orijinal dizgelerin artık korunmasına gerek yoktur [35].

Bir sonek ağacı metin içerisindeki 1,2,3...,n uzunluğundaki tüm öneklerin Şekil2.4'de gösterildiği gibi ağacın üzerine eklenmesiyle elde edilir. Ağaç kullanılarak, verilen dizinin belli bir öneki içerip içermediği hızlı bir şekilde öğrenilebilir. Örneğin CAA araması yapılırken nAA, CnA, CAn gibi hatalı kelimler de aranmış olabilir (n herhangi bir baz anlamında kullanılmıştır) .



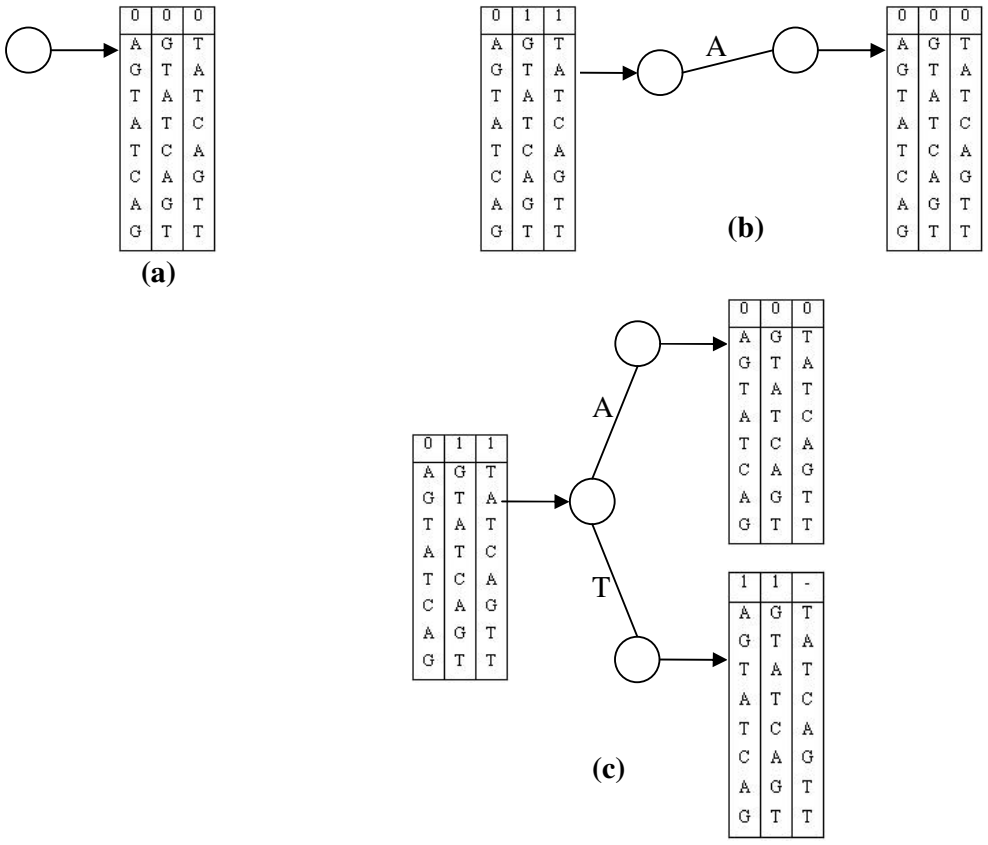
Şekil 2.4: CAGCAAT için sonek ağacı oluşturma.

SMILE'ın tasarımcıları basit bir önek ağacı geliştirmişlerdir. Hesaplamaları kısıtlamak için motif uzunluğunu sınırlamışlardır [36].

2.1.8. Mitra: Uyuşmazlık(Önek) Ağaçları

Bu yaklaşım, birisi Winnower'a diğeri de SMILE'a benzer iki adımdan oluşmaktadır [37]. İlk adımda önek ağacına benzer bir ağaç oluşturulur. Burada kenarlar kısmi önekler olarak gösterilse de kökten düğüme her yol tam bir öneki verir. Şekil 2.5'de görüldüğü gibi, kök ayrıtı başlangıçta boştur ve sabit uzunluklu tüm alt dizgelerin listesine işaret eder. Listeden ilk kenar A öneki olarak gösterilir (b) ve A veya herhangi bir harf ile başlayan tüm alt dizgelerin listesini işaret eder. Sonuç olarak hata sayısı birer artırılarak belirlenir. Bu düğümden devam edilerek daha fazla kenar ağacı belirlenir. Örneğin T olarak adlandırılan kenar, AT ile başlayan herhangi tüm alt dizgeleri işaret eder yada uyumsuzluk sayısı bir arttırılır (c). Eğer tek bir uyumsuzluğa izin verilirse bu uyumsuzluk-eşiğini geçen tüm dizgeler silinmiş olarak işaretlenir, bu dizgelerin tepe bölümündeki uyumsuzluk-sayısı alanında "-" ile gösterilmiştir. Bu dizgeler bir sonraki adımda genişletilmemiştir.

Sonek ağaçlarının aksine önek ağaçları dizilerin bire bir gösterimi değildir.



Şekil 2.5: AGTATCAGTT için önek ağacı. (a)durumunda kök düğümü yalnızdır. (b) ve (c) durumlarında A ve AT önekleri için ağacın genişlemesi gösterilmiştir.

2.1.9. Multiprofiler: Geliştirilmiş Komşuluk Arama

Geliştirilmiş komşuluk arama, verilen dizge içerisinde tam olarak *k* uyumsuzluğa sahip tüm alt dizgeler için arama yapar [38]. Bu k-komşuluk olarak bilinir. Bu komşuluk daha sonra bir karakter dizisi ve karakterlerin belirli bir pozisyonda bulunma olasılığını tutan tekrarlı sözcüklerin aranması ile motif için tarama yapılır. Bunlar komşuluğa göre sayılmış ve numaralandırılmıştır. Sözcükler eğer var iseler en azından belirli bir miktar zamanda işlem görürler. Tüm mümkün sözcükler referans bölümlerine göre toplam uyumsuzluk mesafeleri karşılaştırılır ve en iyi olanları tutulur. Benzer gözükse de en iyi motifin komşuluğundaki olası tüm kombinasyonları aramaktan daha etkilidirler.

2.1.10. Projection: Rastlantısal Özütleme

Bu algoritma özüt için basit *x* pozisyonları kullanır. *l*-alt dizge eğer x pozisyonunda benzer karakterlere sahip iseler beraber bir sepete koyulurlar. Bu örnek güdümlü hesaplamayı tamamlamak, çok fazla zaman tutan bir işlem basamağıdır. Yine de izdüşüm tek başına bu özütlemeye güvenmez [39]. Şekil 2.6'de bir takım motif örnekleri için özütleme listesi ile oluşturma görülmektedir.

Dizi	**C**	**T**	**A**	**A**	**T**	**C**	**A**	**C**	**…**	**A**	**C**	**T**	**A**	**A**	**C**	**…**	**A**	**A**	**G**	**A**	**C**	**G**	**C**

6- Uzunluk Motifler							Motiflerin bulunduğu x noktaları								
C	**T**	**A**	**A**	**T**	**C**	→	**1**	**45**	**57**	**89**	**123**	**142**	**157**	**512**	**680**
T	**C**	**A**	**G**	**A**	**G**	→	**5**	**66**	**73**	**110**	**187**	**197**	**200**	**600**	**655**
A	**C**	**T**	**A**	**A**	**C**	→	**30**	**48**	**450**	**600**	**-**	**-**	**-**	**-**	**-**
G	**G**	**G**	**T**	**A**	**C**	→	**.**	**.**	**.**	**.**	**.**	**.**	**.**	**.**	**.**
G	**C**	**G**	**A**	**T**	**A**	→	**.**	**.**	**.**	**.**	**.**	**.**	**.**	**.**	**.**
T	**T**	**T**	**C**	**A**	**T**	→	**.**	**.**	**.**	**.**	**.**	**.**	**.**		
A	**A**	**G**	**A**	**C**	**G**	→	**.**	**.**	**.**	**.**	**.**	**.**	**.**	**.**	**.**
A	**G**	**A**	**C**	**G**	**C**	→	**18**	**80**	**129**	**.**	**.**	**.**	**1994**	**-**	**-**

Rasgele x noktaları seçilir
Eğer bu noktalardaki motifler benzer karakterler içeriyorsa motif için özüt tablosunda bir indekse oluşturulur.
Bu adım istenilen sayıda tekrarlanır.
Daha sonra herhangi bir motif aranmak istendiğinde özüt tablosunda o motife ait pozisyonlar bulunur ve istenilen işlem yapılır.

Şekil 2.6: 6-Uzunluklu motifler için özüt tablosu.

2.1.11. MoDEL: Evrimsel Hesaplama

Aslında evrimsel hesaplama hemen hemen her yerde uygulanmış genel problem çözme yöntemidir. Algoritma tamamen rasgele yada elle seçilmiş çözüm havuzu (havuz genellikle "popülasyon" olarak adlandırılır) ile işleme başlar, daha sonra bu çözümler operatörler yardımıyla çocuk çözümlere dönüştürülür, son olarak çocuk çözümler değerlendirilir. Ürünlerden yeni bir alt küme elde edilir ve bu alt küme yeni popülasyon olarak adlandırılır [40]. Algoritma adımları tekrarlanarak en uygun çözümler bulunmaya çalışılır.

Bu yaklaşımda giriş popülasyonu rasgele oluşturulmuş motif kümesidir, bunların tümü mevcut durum pozisyonlarına göre sıralanmış haldedir. Daha sonra Evrimsel Hesaplama algoritması aşağıdaki operatörlerden bir tanesini uygular.

(a) Varsayılan motif pozisyonlarını rasgele olarak sağ veya sola yerleştir,

(b) Benzer GC-içeriğine sahip pozisyonları bir alana yerleştir,

(c) Basitçe iki hizayı alır ve onların bir tanesini diğerine göre keyfi olarak tekrar yerleştirir (pencere birleştirme operatörü). Sonra motiflerden bir tanesi tutulur diğeri ise popülasyondan silinir.

Rasgele modifikasyonlar yapılıp tüm motiflerin skorları hesaplandıktan sonra, kopya çözümler elenir. Eğer belirli bir sayıda çevrimden sonra yeteri kadar değişimin olmaması veya özel (kritik) zamanın aşılması olaylarından herhangi biri gerçekleşmemiş ise motif kaydedilir ve algoritma sonlandırılır. Tüm bir araya getirilmiş motifler skorlarına göre sıralanır ve çıktı olarak kullanıcıya gösterilir.

MoDEL'in benzer temel sistemi, operatörleri yerleştirmeler üzerine uygulamaz bunu yerine kelimeler uygular [41]. Bu nedenle mutasyon adımından önce iki seçilmiş pozisyon kelimelere dönüştürülür. Sonra bu iki kelime aşağıdaki 4 seçenekten biriyle değiştirilebilir.

(a) Harfler kelimeler arasında belirli bir olasılık dahilinde birebir değiş-tokuş yap,

(b) Rasgele seçilmiş harfin sağ tarafındaki tüm harfleri değiş-tokuş yap,

(c) Tüm kelimeleri, üst üste binmesi iyi bir şekilde bitinceye kadar kaydır ve yeni kelimeyi oluştur,

(d) Bir kelimeyi sağa veya sola kaydır ve yeni kelimeyi oluşturmak için içerisini rasgele karakterler ile doldur.

Evrimsel hesaplama, ilerleyen zamanlarda motif örneklerini rasgele kaydırma adımlarına sahip yerel bir arama işlemi ile yeniden düzenler. En iyi motifleri bulma skoru, göreceli olarak bilgi içeriği anlamına gelir. Aslında bu yöntem DNA dizilerinde uygulansa da, "dizi başına tek bir oluşum" kavramı nedeniyle destekleyici analiz amacıyla pek kullanılmaz.

2.2. Olasılık Yaklaşımlı Algoritmalar

2.2.1. MEME Algoritması

Beklenen Değer Maksimizasyonu (EM-Expectation Maximization) beklenmeyen değişkenlere bağlı veri içerisindeki maksimum olasılık tahminlerini bulmaya çalışan genel bir yöntemdir [42]. Motif Çıkarma İçin Çoklu EM (Multiple EM for Motif Elicitation), Timothey Bailey tarafından 1995 yılında doktora tezinde öne sürülmüş bir yaklaşımdır [43]. Bu yöntem, özünde proteinler için yazılmış bir maksimizasyon tahminidir.

Varsayımlar ve Veri Yapısı: Bu yöntemde, örnek pozisyonları bilinmeyen değişken olarak kabul edilir. MEME yöntemi dizi başında tek bir rastlantıyı varsayar bu OOPS Modeli olarak bilinir. Bunun haricinde dizi içerisinde daha fazla sayıda örnek olduğunu varsayan modeller de vardır (ZOOPS ve TCM gibi), temelde hepsi özdeştir. Örnek sayıları değişken olabilir veya farklı pozisyonda başlıyor olabilir, algoritma zaman içerisinde bunları araştırarak bulur. Optimal örnek uzunluğu, birkaç yan adımın atlandığı ilave bir beklenen değer maksimizasyonu döngüsü ile örneklerin kısaltılması yoluyla modellenebilir.

Arama Algoritması:

a) Beklenen Değer Adımı: MEME, modelden elde edilmiş (dizilerin taranmasıyla elde edilen) ağırlık matrisindeki aday bölgeleri kabaca bir araya toplar.

b) Maksimizasyon Adımı: Daha sonra beklenen en yüksek değerlere yol açan dizi içerisindeki bu yeni bölgeler ile modeli güncelleştirir.

Bu tip yaklaşımlar, rasgele başlama nokta arama yapıp sonuçta yerel maksimum değerine ulaşırlar ama küresel maksimumu bulmaları zordur. Bu yüzden MEME algoritması, çok faklı başlangıç noktaları ile başlayıp varsayım değerlerine göre sonuçta oluşan ürünleri sıralarlar.

Puanlama: Varsayım değeri, bir motifin rasgele dizilerde bulunma sayısını verir. Bu değer, algoritma yürütülürken algoritmanın beklenen değer hesabının yapıldığı kısmında hesaplanabilir

Deneyimler: MEME algoritması daha çok protein motifleri üzerine yoğunlaştığı için DNA dizileri üzerinde herhangi bir deney yoktur. Birçok algoritma, elde ettiği sonuçları MEME algoritması ile karşılaştırsa bile her zaman kendi çalışmalarının sonuçları daha iyidir. Motif keşfi değerlendirmesinde yine de MEME algoritması diğerlerinden daha iyi sonuç vermiştir. Bu dikkatli parametre seçiminden ve MEME'nin özünde protein motifleri üzerine eğilimli olmasından kaynaklanmaktadır.

2.2.2. Diğer Olasılık Tahmini Yöntemler

Logos olarak adlandırılan yöntem, ağırlık matrislerinin toplamı, bütün son durum ağırlık matrisleri için, prototip olarak hizmet ederi savunur [44]. Tüm prototipler Dirichlet dağılımından elde edilmiş bir örnek değerle başlangıç matrisine katkıda bulunurlar. Dolayısıyla sonucu nasıl etkilediklerine dair dağılım ağırlıklarıyla birlikte verilen bir başlangıç matrisi koleksiyonu olmalıdır. Tüm bu dağılımlar son durum matrisinin bilgi içeriğini maksimize etmek için ilerleyerek örneklenirler. Bir motif modeli tanımlandığında, ona ait dizi de taranır.

Bu yaklaşım matris içerisindeki gizli pozisyonlar saklanmış motifleri kümeler haline getirmek için kullanılır ve biyolojik bakış açısından kabul edilebilir gözükmektedir.

2.2.3. Orijinal Gibbs Yer Örnekleyici

Gibbs bir Markov-zinciri Monte Carlo yaklaşımıdır [45]. Markov zinciri diye adlandırılmasının nedeni her adımdan sonra oluşan yeni sonuçların bir önceki sonuçlara bağlı olmasıdır. Monte Carlo denmesinin nedeni ise seçme işlemi örnekleme temelli olmasından ziyade deterministik olmamasıdır. Bu yaklaşımın MEME algoritmasından farkı, MEME'de beklenen değer maksimize edilirken tek bir örnek seçerken, Gibbs'de her bir örnek belirli bir seçilme olasılığına sahiptir.

Varsayımlar ve Veri Yapısı: Giriş *n* tane diziden oluşan bir kümedir, her küme en az bir tane motif örneğine sahip olmalıdır. Motif uzunluğu *l* ile ifade edilir.

İki tane değişken olarak *l* uzunluklu iki matris kullanılır: *Motifmatrisi* o ana kadarki motiflerin tutulduğu matris, *Arkaplanmatris* geçerli arka plan birleşimini tutar. *Hizala[1..n]* diziler içerisinde bulunduğu varsayılan motiflere ait ofsetleri tutan dizidir. Mesela bir örneğe ait ofset dizisi eğer *hizala[5,5,5,5,5]* ise bu motif tüm dizilerde dizilerin 5. pozisyonundan başlıyor anlamına gelir.

Arama Algoritması:

Başlangıçta *hizalama* dizisi rasgele değerler ile doldurularak motiflerin o pozisyonlarda bulunduğu varsayılır. Daha sonra aşağıdaki adımlar tekrarlanır:

a)Kestirimli Güncelleme Adımı: Bir tanesi hariç hizalama dizisi tarafından gösterilen dizgelerden *motif matrisi* hesaplanır. Bu *z* dizisi sonraki hesaplamalardan tamamen muaf tutulur. Tüm dizilerdeki geçerli motif örneklerini hizalamasına benzer, dizilerdeki diğer tüm *l* uzunluklu dizgeler arka plan, motifsiz olarak dikkate alınır. Bu yüzden *arka plan matrisi* bunlardan hesaplanır. Bu yolla *z* dizisi hariç tüm dizilerden birisi *geçerli motif* diğeri ise *geçerli arka plan* olmak üzere iki matris hesaplanır.

b)Motif Örnekleme Adımı: Bu matrisler verilen dizgelerin motif matrisi ile uyuşup arka plan matrisi ile uyuşmama olasılığı ölçüsü olarak kullanılır. Bu sonuca göre *z* dizisi içerisindeki *l* uzunluğundaki tüm alt dizgeler hesaplanır. $A_x = \frac{P_m}{P_b}$ z dizisi içerisinde hesaplanan her dizge için motif olasılığının arka plan olasılığına oranıdır. Bu numara ağırlık olarak da adlandırılabilir, çünkü algoritma her zaman yüksek oranda dizge seçemeyebilir.

Rasgele (Gibbs) örnekleme $P_x = \frac{A_X}{\sum A_X}$ olasılık değeriyle alınmış bir parça dizge anlamına da gelir.

Puanlama: Değişik motifler birkaç karşılaştırma adımdan sonra bulunur ve en iyileri saklanır. Arama sırasında hesaplanan ağırlıklar daha sonra motif puanlamasında kullanılabilir. Bu yüzden tüm A_x'ler içerisinden en iyi ürüne sahip koşmayı bulmak için arama yapılır. Yada tam olarak $F = \sum_{i=1}^{l} \sum_{j=1}^{4} c_{i,j} \log \frac{q_{i,j}}{p_j}$ formülü ile A_x'lerin logaritmik toplamının en yüksek değerine ulaşılmaya çalışılır. Burada $c_{i,j}$ hizalama dizisi şeklinde gösterilen motif örnekleri içerisindeki *j.* pozisyonundaki *i* nükleik asid sayısını ve q_{ij}, *i.* nükleik asidin *j.* pozisyonda bulunma olasılığını ise p_j ise *j* pozisyonunun olasılığı olarak adlandırılır.

Ayrıntılar: Eğer motif uzunluğu *l* bilinmiyor veya kullanıcı tarafından belirlenmemişse algoritma optimal bir tanesi için arama yapabilir. Bu durumda mümkün olan uzunluk aralığının denenmesiyle birçok yeni başlangıç yapılır. Yine de bunları karşılaştırmak ve optimal bir tanesini bulmak, değişken uzunluklu dizgelere göre F puanını normalize etmektir. Aksi halde puan daha uzun dizgeleri tercih eder.

2.2.4. Neuwald'ın Motif Örnekleme Yaklaşımı

Varsayımlar ve Veri Yapısı: Giriş *k* motife sahip olduğu varsayılan uzun bir dizidir. Eğer kullanıcı birden fazla dizi verirse bunlar ard arda birleştirilir. Sonucun *e* örnek içermesi beklenir. Tek bir motif matrisi kullanılmaz, bunun yerine çoklu motiflerin arandığı *motifmatris[0..k]* şeklindeki matris kümesi kullanılır. Bu nedenle *hizalama[0..k,1..max]* şeklindeki iki boyutlu bir dizi haline gelir. *max* değişkeni mümkün örnek sayısının teorik üst sınırıdır.

Tamamen yeni bir konsept olan bu yaklaşımda, her *motif matrisinin* bir *motif maskesi* vardır [46]. Bitsel yapıdaki bu matris, ağırlıkların hesaplanması sırasında matristeki hangi

pozisyonların önemli olduğunu ortaya koyar. Bir dizge ile matris arasındaki benzerliğin hesaplanması işleminde de aynı yöntem kullanılır.

Algoritma: Temel algoritma, dizi kümesinin yerine tek dizi ve çoklu matrislerin yerine tek matris üzerinde işlem yapıldığı durum haricinde Gibbs ile benzerlik gösterir. Bu yüzden başlangıç aşamasında, tüm *k* motif matrisleri için dizi içerisinde *l* uzunluğundaki *e* örneklerini rasgele seçer ve daha sonra bunları *hizala[1..k,1..e]* dizi içerisinde saklar. Algoritma aşağıdaki adımları içerir.

1. Kestirimli Güncelleme Adımı: Gibbs'e benzer olarak örneklerin pozisyonlarından motif matrisi hesaplanır. Burada matris yerine *k* motiften *e* örnek şeması oluşturulur.

2. Motif Örneklendirme Adımı: Dizi içerisinden bir alt *s* dizgesi seçilir. Bu motif matrisleri puanlamaya sokulur ve bunlardan rasgele ağırlıklı bir tanesi seçilir. Bu *motif matris[0]* , arka plan dağıtımı olarak kabul edilebilir. *s* alt dizgesi yeni örneklenmiş matrisin bir durumu olarak sayılır ve hizalamalara eklenerek sonuç kaydedilir.

3. Sütun Örnekleme Adımı: Zengin bilgilere sahip pozisyondaki motifler diğerlerinden daha önemlidir, bu yüzden algoritma benzerlik hesabı için sütunların bir alt kümesini kullanır.

4. Yakın-Optimum Örnekleme Adımı: İlk üç adımın sabit sayıdaki ilerletilmesinden sonra en iyi motifler tutulur. Bu matris için dizi örnekleme işlemi sütun örnekleme işlemi adımı olmadan ve *e*'nin daha iyi değerleri ile birlikte devam eder.

Puanlama: Örnekleme aşamasında bilgi içeriği MAP puanı ile yeniden yerleştirilir.

2.2.5. AlignACE

Motif örnekleyicilerdeki gelişmeler AlignACE tarafından daha uygun gerçek biyolojik örneklerle genişletilerek devralınmıştır [47]. Maya üzerindeki veriler işleme ışık tuttu. Aynı zamanda motiflerin biyolojik önemini değerlendirme işlemini basite indirgeyen ve benzer motifleri kümeleştiren araçlar ile donanmıştır.

Algoritma: Araştırma sürecinde oldukça küçük değişimler meydana geldi. Nükleik asitlerin ihtimallerindeki öncelikler maya genomlarının değerlerinde sabitlenmiştir. Dizi başına tek bir motif varsayımı genellikle destekleyicilere göre geçerli değildir, algoritma bu nedenle çoklu motife adapte edildi. Takip eden adımlarda aynı motifin birkaç kez bulunma problemini gideren herhangi bir oluşum saptandığında buna ait güçlü pozisyonlar daha sonra örnekleme adımlarında kullanılmazlar.

2.2.6. ANN-Spec: Yapay Sinir Ağları Kullanmak

Hiza tespiti taramasında en eski yaklaşımlardan birisi olan bu yaklaşım bir algısal öğrenmedir [48]. ANN-Spec, Yapay Sinir Ağı kullanmasına rağmen orijinal Gibbs örnekleyici ile daha fazla benzerlik gösterir: hizalanan veya hizalanmayan alt dizgeleri sınıflandırmak için bir perceptron kullanılır bu yüzden Gibbs'deki matris ağırlıkları ile perseptron ağırlıkları birbiriyle uyuşur.

Varsayımlar ve Veri Yapılar: Motif içeren diziler ve arka plan dizileri verilir. Bir perceptron puanlama ve sınıfları hizalanan veya arka plan olarak sınıflandırma için kullanılır.

Algoritma: Perceptronların ağırlıkları dizilerde rasgele hizalanmış şekilde örneklenmiş olarak göstermek için ayarlanır. Daha sonra belli bir sayıda iterasyona ulaşıncaya kadar aşağıdaki adımlar tekrarlanır:

1. Motif Örnekleme Adımı: Gibbs'deki h_j olasılığı burada doğrudan kullanılmaz. Bunun yerine exp^{h}_{j} fonksiyonu uygulanır. Sonuç her alt dizge için kaç *k* hizada örneklendiğine dair bir ağırlıktır.

2. Ağırlık Güncelleme Adımı: Hedef seçilen sitelerin logaritmik komşuluğudur. Daha sonra ağırlıklar, $\Delta\Omega = \eta\left(\frac{\Delta U^{*}}{\Delta\Omega}\right) - \lambda\Omega$ formülü kullanılarak güncellenir. Burada Ω ağırlık vektörü, η öğrenme oranı, U^{*} amaç ve λ ise bozulmayı temsil eder. Bu yüzden değişim, örnek hizalarının eski ağırlık ile bölünüp, bozulma faktörü ve öğrenme oranı tarafından düzeltildikten sonraki logaritmik komşuluktur. Temel formül perceptron öğrenmesi için standarttır. Bu algoritma, Gibbs'den bazı yönleriyle farklıdır; burada ağırlıklar yeni seçilmiş hizaların doğrultusunda ayarlanmıştır. Bunlar doğrudan hizaları yansıtmaz.

Puanlama: Burada amaç logaritmik komşuluktur ve dolayısıyla bu puan ile uygunluk gösteren bir parametredir. Bu durumda, bir genom içerisindeki benzer hiza sayısının dizi içerisindeki olası tüm hizaların toplamına bölünmesiyle h_j Boltzmann olasılığı elde edilir. Eğer motifler kendi aralarında iyi benzerlik gösteriyorlarsa puanı arttırırlar. Eğer motifler birbiriyle benzerlik göstermiyorsa bir sonraki adımda çözüm kümesinden atılır.

Deneyimler: Rasgele diziler oluşturulmuş mutasyona uğramış motifler içerlerine uygulanmıştır. Gibbs bunun üzerinde ANN-Spec'a göre birazcık daha iyidir ama MEME [43] ve Consensus'un [33] gerisinde kalır.

2.2.7. Thijs'in Motif Örnekleme Yaklaşımı

Varsayımlar ve Veri Yapısı: Algoritma 2001'de önerilmesine rağmen 1993'deki orijinal Gibbs hizalama örneklemesinin [45] devamı niteliğindedir [49]. Daha sonraki çalışmalarda olduğu gibi, Thijs "dizi başına bir tane motif" sınırlamasının uygun olmadığını düşünüp bunu kaldırdı, bu nedenle sıfır yada c_{max}(maksimum örnek sayısı) dizi içerisinde görülebilir. Bu yüzden pozisyonların listesi her dizi için tekrardan çoklu kayıtlara sahiptir.

Γ dağılımı, her kopya sayısına bir olasılık değerinin yüklendiği dizi olarak kaydedilebilir. Her alt dizge için arka plan olasılığı hesaplanmalı ve algoritma başlamadan önce kaydedilmelidir.

Algoritma: Arka plan olasılıkları hesaplandıktan sonra hizalar rasgele numaralar ile ayarlanır ve bunlarda Γ kopyalarının dağılımı hesaplanır. Aşağıdaki adımlar maksimum iterasyon sayısına ulaşıncaya kadar tekrarlanır:

1. Kopyaları Örnekleme Adımı: Her dizi için, beklenen c_k kopya sayısı Γ'den örneklenir.

2. Kestirimli Güncelleme Adımı: Orijinal Gibbs'de olduğu gibi, pozisyonlardaki bir z dizisine ait tüm kayıtlar silinir, dizilerdeki varsayılan hizalar indirgendiğinden geriye sadece c kopya kalır. Pozisyonlar kullanılarak motif matrisi güncellenir.

3. Motif Örnekleme Adımı: z dizisi içerisindeki tüm dizgeler için motif olma olasılığı hesaplanır. Motifin arka plan olasılığına bölünmesiyle, tüm alt-dizgeler puanlanır. Bu dağılımdan c durumlarını örnekle ve hizalama dizisine ekle. Γ'yi güncelle.

Puanlama: Değişik çıkışlar ölçülebilir: Gibbs'de olduğu gibi arka plana bağlı bilgi içeriği, saf motiflerin bilgi içeriği, logaritmik komşuluk. Bu değer, şu logaritmik olasılıkların toplamıdır: arka plan tarafından üretilmiş dizinin olasılığı, gözlenen motiflerin maksimum C kopyasının olasılığı, olası tüm kopyaların toplamı.

2.2.8. Diğer Gibbs Örnekleyicileri

Bioprospector: Bioprospector Orijinal Gibbs örnekleyicisinden aşağıda verilen dört nedenden dolayı faklılık gösterir [50]:

1.Markov Arka Plan Modeli: Bölümler üçüncü sıra Markov modeline bağlı olarak puanlanmıştır.

2.Eşik Örnekleme: Örnekleme adımında bölüm seçiminde iki tür eşik kullanılır: Yüksek eşik değerli tüm alt dizgeler, doğrudan doğruya seçilir, bunlarla birlikte düşük eşik değerli tüm

alt dizgeler örnekleme adımdan atılır. Örnekleme sadece bu iki eşik değerini arasındaki puana sahip alt dizgeler üzerinde uygulanır.

3. Özgül Motif Modelleri: Kullanıcı iki sabit boşluk belirtebilir. Daha sonra örnekleyici tersi kopyalanmış ve normal olmak üzere, ayırıcıya sahip iki faklı matris tanımlayabilir ya da bir matrisi iki defa uygulayabilir.

4. Düzeltilmiş Motif Puanlama Şeması: Bir motifin bilgi içeriği birkaç örnek içerisinde bulunsalar bile motifler için iyi olabilir. 150 örnekli bir motif genellikle düşük seviyede bilgi içeriğine sahip olsa bile biyolojik olarak daha anlamlıdırlar.

Algoritma maya ve bakteri verileri üzerinde kullanılmış fakat diğer canlı dizileri üzerinde karşılaştırma yapılmamıştır.

MDScan: Bu algoritma bütün uygulamalara adapte olan orijinal Gibbs örnekleyicisinin hafifçe düzeltilmiş halidir [51]. Belirginleştirilmiş diziler motif içermeye daha fazla elverişlidir, bu yüzden MDScan diziler içerisindeki uyumsuzluğu aramada tüm l uzunluklu alt dizgeleri başlangıç noktası olarak kullanır. Benzer olarak verilen tüm alt dizgeler bir kelime olarak verilir ve daha sonra kaydedilir. Bunlardan bir matris hesaplanır. Arama için başlangıç noktası olarak verilen bu matris, Motif örnekleyicinin ilk kestirimli güncelleme adımı olarak kullanılır. MDScan bu yüzden Örnek güdümlü numaralandırma ve Gibbs örnekleyicisi arası bir yöntemdir. Sonuç olarak MDScan belirsiz sayıda örneğin olduğu veriler üzerinde işlem yapan AlignACE, Consensus yada Bioprospector yöntemlerinden daha iyi puanlama yaptığı bilinir.

Co-Bind: Co-Bind bitişik motiflerin bir boşluk ile ayıran iki çift modelden geliştirilen bir motif modelidir [52]. Gibbs örnekleyici kullansa da farklı bir kestirimli güncelleme adımına sahiptir: bu adım Ann-Spec ile benzerlik gösterir. Yeni seçilmiş örneklerden doğrudan matris hesaplaması yapmaktansa yüzdelik indirgeme yapılır. Puanlama modeli bir gene bağlı iki etken için açıkça olasılık hesaplaması yapar. Aralarında çok büyük farklar olmasa da Co-Bind Bioprospector algoritmasından daha önemli motifleri bulur.

Ampheta Meme: Arka plan için Markov modeli kullanan bir başka Gibbs örnekleyici algoritması Jim Kent tarafından yazılmıştır [53]. Bu puanlama esnasında arka plan dizilerini kullanır ve motif uzunluğunu artırabilir.

SeSiMCMC: Dizi başına motiflerin 1 yada 0 olasılığını arayan bir Gibbs örnekleyicidir [54]. Bu yaklaşım örnekleme işleminde motif uzunluğunu optimize eder ve dizileri tarama yaparak bulunan motifleri geliştirir. Ayrıca puana bağlı olarak göreceli bir bilgi içeriği kullanır. Bu algoritma bakteriler üzerinde uygulanmıştır.

2.2.9. GMS-MP

Daha önceki diğer birçok yaklaşım ağırlık matris modelinin geliştirilmesi esasına dayanır. Genellenmiş Ağırlık Matrisinde(GAM), motif içerisindeki tüm harfler bir pozisyonda olma olasılığına sahiptirler, fakat verilen gerçek bir nükleotid diğeri ile ilişkilidir. Yaklaşımı önerenler bunun diğer herhangi bir karmaşık algoritmadan daha az parametreye ihtiyaç duyduğunu savunurlar [55].

Varsayımlar ve Veri Yapısı: GAM iki tür matris içerir: bir tanesi olağan nükleotid ağırlık matrisi, diğer uygulama yapılan matris üzerindeki pozisyonları belirlemede kullanılan (x,y) özelliğine sahip bazı denükleotid matrislerdir. Denükleotid matrisleri iki nükleotidin olası tüm kombinasyonlarının listesini içerir ve x ile y noktalarındaki görülme olasılığını verirler. Örneğin (4,5) pozisyonunu uzunluğu 6 olan motif içerisinde ilişkilendirildiğini varsayılsın. Burada 1,2,3,6 pozisyonları alışa gelindiği gibi aynı kalır, fakat 4 ve 5 pozisyonları için 4 ile 5 pozisyonlarının olası tüm kombinasyonlarının ve göreceli olasılıklarının listelendiği ayrı bir matris oluşturulur.

Orijinal motif örnekleyicideki ikinci değişim önceliğin tanımlı olmamasına benzer kullanıcı tanımlı hiçbir parametre içermemesidir.

Algoritma: Bu yaklaşımda örnekleme adımı öncelik için bir değer seçme vardır ve sadece normal ağırlık matrisini örneklemez fakat GAM'in tüm ağırlıklarını örnekler.

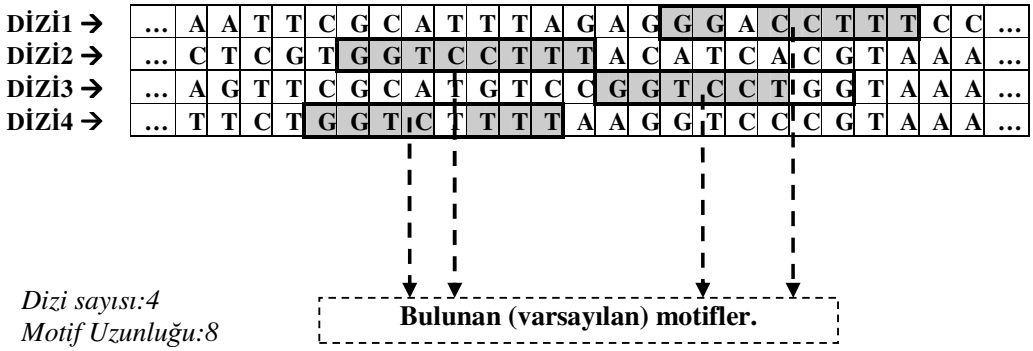
Deneyimler: Tüm örnekleyiciler bir dizi üzerinde yürütüldü ve çok iyi ilişkilendirilmiş gömülü motifin 1 ve 2 tane pozisyonunu verdi, GAM örnekleyici gerçek motife benzer matrisi tutarken orijinal motif sayılarını yakın sonuçlar verdi. Yine de aralarındaki farklar tam olarak kestirilmemektedir.

3. MOTİF ÇIKARMADA YAKLAŞIMSAL YÖNTEMLER

3.1. Dizi Hizalama

Dizi hizalama, diziler içerisindeki benzer sıralara sahip karakterleri veya karakter örüntülerini arama yaparak, protein veya DNA dizilerinin iki veya çoklu karşılaştırması işlemidir. Dizi çeşitliliği arttığında dizilerdeki benzer karakterlerin optimal hizalaması zorlaşır. Çoklu dizi hizalama olası hiza sayısı büyüdüğünde çözüm karmaşıklığını arttırır [56].

Dizi hizalanmada iki türlü yaklaşım vardır. Küresel hizalama çoğu veya tüm dizileri değerlendirir buna nazaran yerel hizalama faklılık gösteren bölgeleri yok sayarak sadece birbirine çok benzeyen bölgeleri hizalar. Profil analizi ve blok analizinin ilk adımı çoklu dizi hizalamadır. Şekil 3.1'de dizi hizalama için basit bir örnek gösterilmiştir.



Şekil 3.1: Dizi hizalama işlemi.

Hizalanan motifler tam manasıyla benzerlik göstermeyebilir. Ayrıca aynı noktada bulunmama olasılığı yüksektir. Tüm algoritmalar belirli bir tolerans ile motif değerlendirmesi yaparlar. Bazı uyumsuzluklar dikkate alınmayabilir.

3.2. MEME Algoritması

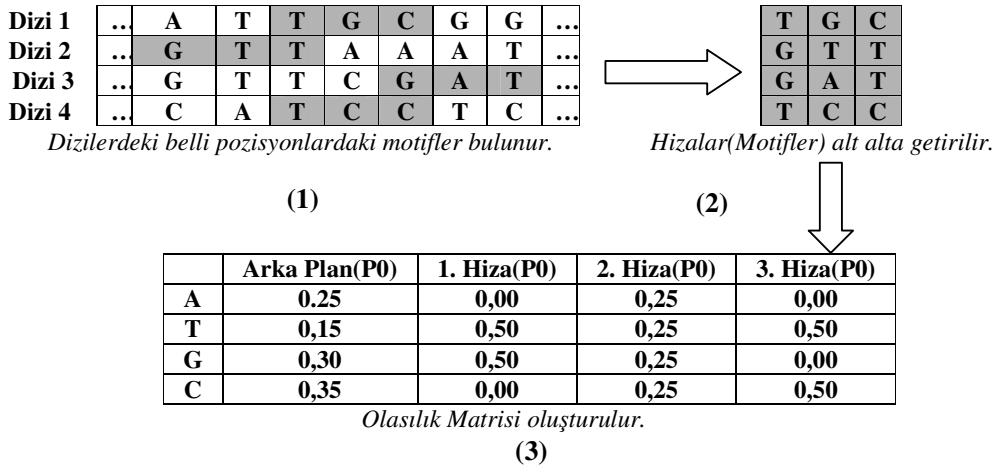
3.2.1. Giriş

MEME, ilişkilendirilmiş DNA ve protein dizilerindeki çok iyi saklanmış bölgeleri, motifleri yada grupları tanımlamak için kullanılan istatistiksel bir yöntemdir [43]. Bu yaklaşım, DNA ve protein dizilerindeki hizalanmamış motifleri tanımlamak için beklenen değer maksimizasyonunu kullanan bir teknik geliştirmiştir. Algoritma öğrenmeli ya da öğrenmesiz algoritma olarak çalışır. Motif sayısı ve uzunluğu, tüm diziler için motif başlangıç pozisyonları giriş bilgisi olarak kullanılır.

3.2.2. Beklenen Değer Maksimizasyonu

Beklenen Değer Maksimizasyonu (BDM), MEME algoritmasının temelidir. BDM algoritması tekrarlanarak çözüm yapan iki adımlı işlemdir. İterasyon başlamadan önce motifler için başlama pozisyonları tahmini yapılır.

Başlangıç motifi bir sütundaki bazların frekanslarının matrisi ile ifade edilir. Bu matrise "*olasılık matrisi*" denir. Bu matris; her bir bazın sütun içersindeki frekanslarının ayrı ayrı hesaplanmasıyla oluşturulur. Hesaplanan bu değerler yüzde olarak gösterilir. Motif uzunluğu 3 ve dizi sayısı 4 olan bir işlem için Şekil 3.2'de bir matris oluşturma açıklanmıştır.



	Arka Plan(P0)	1. Hiza(P0)	2. Hiza(P0)	3. Hiza(P0)
A	0.25	0,00	0,25	0,00
T	0,15	0,50	0,25	0,50
G	0,30	0,50	0,25	0,00
C	0,35	0,00	0,25	0,50

Şekil 3.2: Olasılık matrisi oluşturma.

Tahmin ve maksimizasyon adımları, diziler içerisindeki motiflerin olası pozisyonlarının olasılıklarını hesaplamak için sırayla uygulanır ve algoritmanın durdurma şartları oluşuncaya kadar motif modelini keşfetmeyi sürdürür. Tahmin adımı belirlenen motifi, motifin tüm dizilerin herhangi bir pozisyonundaki bulunma olasılığını hesaplamak için kullanır. İkinci adımda maksimizasyon, verilen motifin olasılığını tekrar hesaplar. Bu işlemler motif üzerinde çok az değişim oluncaya kadar devam eder.

Algoritma aşağıdaki değişkenleri kullanır:

S L uzunluğundaki hizalanmamış diziler, veri kümesi(*dizi1,dizi2,dizi3*..)

W motif uzunluğu

z motifin S_i dizinde j pozisyonunda bulunma olasılığının matrisi

ρ bu matriste c karakterinin k sütununda bulunma olasılığı tutulur. Burada c karakteri DNA dizilerinin yapısında bulunan A,C,G,T bazlarında biri olabilir.

ε *epsilon* değeri

Algoritma:

1) EM (S, W) {
2) ρ için ilk değer ve başlangıç noktası seç
3) do {
4) ρ matrisinden z matrisini hesapla→Ölçme adımı
5) z matrisinden ρ matrisini tekrar hesapla →Maksimizasyon adımı
6) } until (ρ 'nin değişim miktarı < ε)
7) ρ , z değerlerini göster.
8) }

3.2.3. MEME Algoritmasının Yapısı

MEME algoritması aşağıdaki yeni gelen özellikler ile temel maksimizasyon yaklaşımından geliştirilmiştir:

a) Motiflerin küresel (tüm dizilerde) bulunma olasılığını artırmıştır. Nükleik asit ve amino asitlerin alt dizilerini başlangıç noktası girişi olarak alır. BDM yerel optimal maksimumu bulmayı garanti eder.
b) Dizilerdeki çoklu motif oluşumunu destekler. Gürültü oranı fazla olan veri dizilerinde dahi, BDM yaklaşımındaki dizi başına tek bir ortak motif bulunma olasılığını değiştirir ve arttırır.
c) Tek bir dizi içerisinde birbirinden farklı ortak motif keşfedebilmeye imkan tanır. Bunu, BDM tarafından keşfedilen ortak motifin rastlantısal olarak eleme ve tekrar BDM yaklaşımı ile motif bulma ile yapar.

Algoritma şu adımlardan oluşur:

```
1) MEME(S, W, NSITES, PASSES) {
2)     for i = 1 to PASSES {
3)         S içerisindeki tüm alt diziler için{
4)             Bu alt diziden elde edilmiş bir başlangıç noktasıyla BDM işlemini
                bir iterasyon yürüt
5)             yüksek olasılıklı ortak motif modelini seç
6)             bu modelden üretilmiş başlangıç noktasını yakınsamak için
                BDM'yi yürüt.
7)             ortak motif modelinin yakınsağını göster
8)             veri kümesinden ortak motif modelini sil.
9)         }
10)    }
11) }
```

Algoritma motif tekrarını önlemek için bulunan ortak motif örneğini diziden çıkarır.

3.3. MDGA: Genetik Algoritma Kullanarak Motif Keşfi

Bu yaklaşımda, birbirine benzer genlerin hizalarını verimli bir şekilde tahmin edilmesi hedeflenmiştir [57]. MDGA'da bireyler , benzer özellik gösteren farklı diziler içerisinde hizaların olası başlangıç noktalarının kümesi olarak gösterilir. Her bir birey için uygunluk değeri, o bireyin ağırlık matrisinin her bir sütunundaki toplam bilgi içeriğinin toplanmasıyla elde edilir. Algoritma hizalamalardaki düşük benzerliğe sahip bireyleri cezalandırır dolayısıyla bu benzerliği yüksek bireyleri seçer.

Algoritma Yapısı:

1)Popülasyonun Oluşturulması: Popülasyondaki bireyler ikilik sayı sistemiyle kodlanmıştır. Her birey 16 bit ile kodlanır. Burada bireyler aslında motiflerin başlangıç noktalarının kodlanmış halidir. Popülasyon, ikilik kodla rasgele kodlanmış bireyler ile oluşturulur. Popülasyon boyutu başlangıçta tanımlanır. Evrim sırasında bu sayı değişmeden sabit kalır.

2) Uygunluk (Fitness): Algoritma uygunluk değeri olarak motiflerin benzerlik oranları ölçüsüne bakar. Bu değer bilgi içeriği olarak bilinir. Bu yaklaşım tek bir sütundaki bilgi içeriğini bulmak için $IC = \sum f_b \log_2 \frac{f_b}{p_b}$ formülünü kullanır. Burada f_b, b nükleotidinin o sütundaki frekansını ve p_b aynı nükleotide ait arka plan frekansıdır. Örneğin eğer 4 motif bulunmuş ise p_b=1/4= 0.25, eğer 8 motif bulunmuşsa p_b=1/8= 0.125 olur. Bu tek bir kolondaki bilgi içeriğini bulur. Bu işlem tüm sütunlar için ayrı ayrı yapılır ve sonuçta kolonlarda oluşan bilgi içerikleri toplanır ve motif uygunluğu bulunur. Motif uygunluğu; $uygunluk = \sum_{i=1}^{w} IC_i$ formülü yardımıyla hesaplanır. Formüldeki *w* motif genişliğini gösterir.

Eğer baz frekansı 0 ise hesaba katılmaz. Bu yaklaşımda baz frekansları için sahte sayılar kullanılır. Bu şekilde baz frekansları ve arka plan olasılığı değişik bir form alır. Sütundaki baz frekansı $f_b^{'} = \frac{c_b + d_b}{N - 1 + D}$ formülüyle, arka plan olasılığı $p_b^{'} = \frac{c_{0b} + d_b}{S + D}$ formülüyle hesaplanır. Formüllerde bulunan c_b ve c_{ob}, b nükleotidinin sütundaki tekrar sayısını, d_b, b sahte sayısını, *N* dizi sayısını, *S* arka planda bulunan tüm sayıların toplamı ve *D* ise tüm nükleotidlerin sahte sayılarının toplamını göstermektedir. Algoritma çalıştırıldığından bir süre sonra optimal olmayan yerel optimal değerine kilitlenebilir. Bu sorun, motiflerin başlangıç noktalarının küçük değerlerde sağa veya sola kaydırılması ile aşılmaya çalışılmıştır.

3)Seçme: Tüm iterasyonlarda çaprazlama ve çocuk birey üretimi için belli olasılıklar ile 2 örüntünün seçilmesi gerekir. Buradaki her bir birey sıfırdan farklı bir seçilme olasılığına sahiptir. Bireylerin seçiminde, bireyin seçilme olasılığı bireyin popülasyondaki tüm bireyleri ile normalize edilmiş uygunluk değeri kullanılarak, Roulet tekerleği mekanizması ile yapılır.

4)Çaprazlama ve Mutasyon: Tek veya çift noktalı çaprazlama yapılabilir. Bitsel mutasyon yapan bir operatör kullanılır. Uygula programında kullanıcı mutasyon ve çaprazlama olasılığını kendisi belirleyebilir.

5)Tekrar Yerleştirme Stratejisi: Her yeni nesilde popülasyonun yarısına eşit sayıda çocuk birey üretilir. Bu yeni bireyler popülasyona eklenir. Tüm toplumdaki birey sayısının 1/3'ü kadar en kötü birey toplumdan elenir.Örneğin 100 kişilik bir popülasyonda 50 yeni birey üretilir. Bu bireyler topluma eklenir toplumdaki birey sayısı 150 olur. Bunların 50 en kötü olanı toplumdan atılır.

Algoritma Yapısı:

1. PARAMETRELERİ AYARLA
   a. W motif uzunluğunu belirle
   b. G maksimum iterasyon numarasını belirle
   c. S kaydırma aralığını belirle
   d. Çaprazlama Stratejisi Belirle
2. POPÜLASYON DÜZENLEME

Rasgele ikilik değerler ile birey oluştur ve topluma ekle (bireyler motif başlangıç noktalarını gösterir).

3. YÜRÜTME

Tüm bireyler için

{

Do Loop (-S<= i <=S)

{

Tüm pozisyonları *orijinal_pozisyon* + *i* değerine eşitle;

Diziler içresindeki *w* uzunluğundaki motifleri yeni pozisyonlar yardımıyla bul.

Tüm kolonlardaki uygunluk değerini bilgi içeriklerini toplayarak hesapla

Eğer *geçerli_uygunluk* > *maksimum_uygunluk* ise →

maksimum_uygunluk = *geçerli_uygunluk* yap

}

RETURN *maksimum_uygunluk*

}

4.ÇAPRAZLAMA-MUTASYON

iterasyon_numarası = *G* oluncaya kadar tekrarla

{

Roulet çemberi yardımıyla iki ata birey seç;

Çaprazlama stratejisi yardımıyla bireyleri ÇAPRAZLA;

Çocuk bireyleri MUTASYON işleminde geçir.

Yeni bireyleri topluma ekle;

Kötü bireyleri toplumdan çıkar;

}

5.ÇIKIŞ: Başarılı bireylerin ortak motifini göster;

3.4. FMGA: Genetik Algoritmayla Motif Bulma

Protein sentezini başlatan mRNA hücre çekirdeğindeki DNA üzerinden kopyalanır. DNA sarmalının bir parçası açılır ve mRNA üzerinde açılan bu parça kopyalanır. mRNA ilk başta sadece kopyalama işlemeni başlatan transkripsiyon faktörü (TF) adındaki bir başlatma faktörü içerir. DNA dizilerinde sıklıkla rastlanan TF'lerden bir kaçı şunlardır: TATAbox [TATAAA], BRE [(G/C) (G/C) (G/A)CGCC], Inr [TCA(G/T) T(T/C)], ve DPE [(A/G) G(A/T) (C/T) (G/A/C)] olarak bulunmuştur [26]. Bu protein başlangıcını tetikleyen motifleri bulmak evrimsel yaklaşımların başlıca amaçlarındandır.

FMGA da motif keşfetmede kullanılan evrimsel yöntemlerden bir tanesidir. Önerilen bu yaklaşım Gibbs Örnekleyiciden daha kesin ve hassas sonuç bulur bunun yanı sıra MEME algoritmasından daha az zaman harcar [26].

Yöntem: Bu yaklaşım toplam uygunluk puanlama fonksiyonu ve optimal motifi bulmak için genetik algoritma kullanır. Genetik algoritmanın alt yapısındaki operatörleri kullanarak motif keşfi yapar.

3.5. Genetik Algoritmayla Zayıf Motiflerin Bulunması

Hizalama için logaritmik yapıda olan $I_{align} = \sum_{i=1}^{l} \sum_{b \in \Sigma} f_{b,i} \log_k \frac{f_{b,i}}{p_b}$ formülünü kullanır. Bazen farklı motif parçaları aynı hizalama skoru verir. Bu yaklaşımda iki hizalama arasında karar vermek için $I_{align} = \sqrt[\alpha]{\left(I_1^{\alpha} + I_2^{\alpha} + I_3^{\alpha} + ... + I_L^{\alpha}\right)/l}$ den faydalanılır [58].

Bu algoritma DNA üzerinden elde edilen alt dizgeleri benzerlikleri doğrultusunda kümelere böler. Her bir DNA dizisinden bir veya daha fazla alt dizgeden oluşan kümeler

meydana gelir. Bunlar minimum uzaklık alt dizgeleri olarak adlandırılır. Bir tane X adında uzlaşı bireyi geldiğinde bunu bir kümeye ekler. Ekleme işleminde ilk önce en fazla küme en yukarıda olacak şekilde diğer kümelerdeki bireyleri bu kümedeki elamanların altı yazar. Bu şekilde kümeledikten sonra uzlaşı motifini sırayla oluşan kümlere ekleyerek skorlarını hesaplar. Sonuçta bu X bireyini en fazla skora sahip kümeye ekler. Çaprazlama ve mutasyon aşamaları diğer genetik algoritmalar ile benzerlik gösterir.

4. GENETİK ALGORİTMA KULLANARAK MOTİF KEŞFETME

4.1. Genetik Algoritma

4.1.1 Genetik Algoritma Tekniği

Genetik Algoritmalar (GA) bilgisayar bilimlerindeki ilerlemeler sonucu ortaya çıkan bir evrimsel hesaplama tekniğidir. Michigan Üniversitesinden John Holland 1960'ların başında yaptığı çalışmalarda genetik algoritma kullanmaya başlamıştır [59]. GA'larda amaç: doğal adaptasyon işleminin anlaşılmasını geliştirmek ve doğal sistemlerin özelliklerine benzer yapay sistemler tasarlamaktır. Bu teknik canlılardaki genetik sistemden esinlenerek geliştirilmiştir. Günümüzde genetik algoritma çeşitleri de artmış. Bu da optimal sonuçların alınması sürecinin kısaltıp, yöntemin ekonomikliği artırmıştır.

Genetik algoritmalar kullanılarak günümüzde biyoloji, ekonomi, kimya, sosyoloji gibi birçok alanda karşılaşın çeşitli problemlere çözüm getirilmeye çalışılmaktadır. Bazen de problem çözümlerini optimal yapmak için kullanılır. Dolayısıyla bir optimizasyon tekniği de denilebilir. Problemdeki parametre sayısının fazla olması ve her bir parametre değeri değişimi için yeniden hesaplama yapmak uzundur ve çözüm işleminin karmaşıklığını arttırır. Genetik algoritma ile en iyi veya en iyiye yakın çözümler bulmak daha düşük algoritma karmaşıklığı ile olasıdır. Çünkü olası tüm çözümler sırayla denenmez. Uygun çözümler zaman içerisinde evrimleştirilir ve geliştirilerek en iyi ürün alınmaya çalışılır.

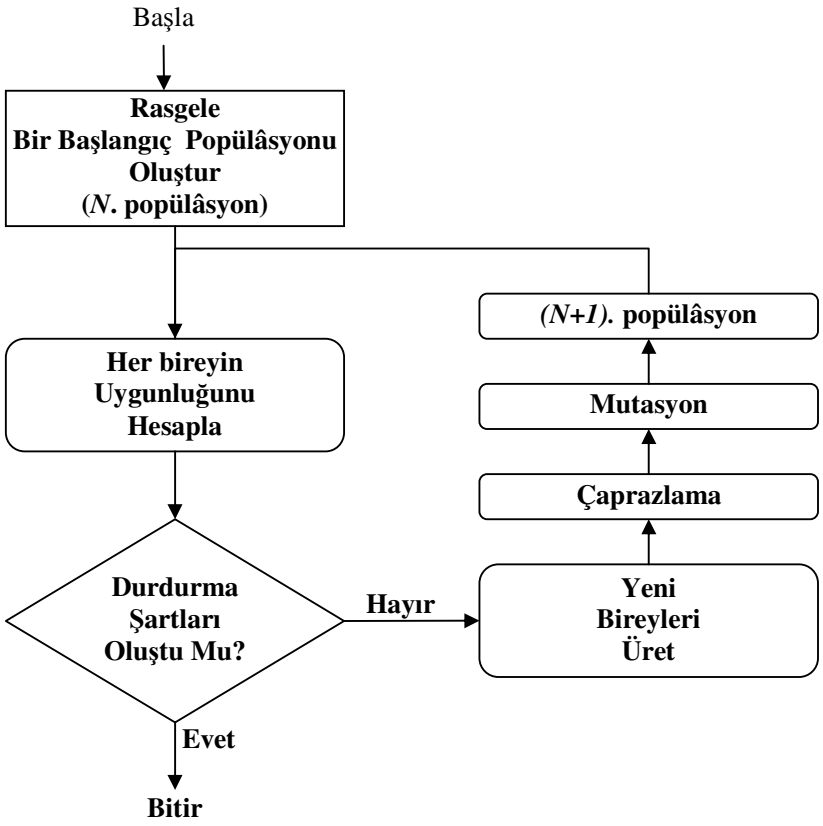
Bu yaklaşımda probleme ait bir grup çözüm yer alır. Bu çözüm havuzu "*popülasyon"* veya "*toplum"*, toplumdaki her bir elaman ise "*birey"* olarak adlandırılır. Popülasyon boyutu ve birey uzunluğu problemden probleme değişir. Çözüm havuzundaki her bir birey ikilik sistemde kodlanır.

Optimal sonuçların bulunması için bireyler bir dizi genetik işlemden geçirilir. Bu işlemlerin sonucunda oluşan bireyler "*çocuk bireyler"* olarak adlandırılır. Popülasyona bu yeni oluşan bireyler de eklenir ve popülasyon yenilenir. Yeni eklenen bireyler dahil olmak üzere tüm bireyler belirli bir uygunluk hesaplama işleminden geçirilir. Bunun sonucu olarak yeterince uygun bulunmayan bireyler sistemden elenir. Eleme kriterleri problem çözücü tarafından belirlenir ve her problem için farklı değer alabilir. Sonuçta sistemde kalan bireyler, yüksek uygunluğa sahip bireyler olarak kabul edilir.

Algoritmanın sonlanması için bir takım şartların oluşması gerekir. Durdurma şartları bazen adım sayısı bazen uygunluk eşik değerinin aşılması veya yeterince uygun bireye ulaşmak olabilir. Sonuçta oluşan çözümler kullanıcıya gösterilir.

4.1.2 Genetik Algoritmanın Yapısı ve Çalışma prensibi

Temel bir genetik algoritmanın yapısı Şekil 4.1'de gösterilmiştir.



Şekil 4.1: Temel bir Genetik Algoritmanın yapısı.

Temel bir genetik algoritma şu adımlardan oluşur:

Adım 1: Problemin ikilik olarak kodlanması.

Adım 2: Rasgele bireyler(n adet kromozom) üret ve başlangıç popülasyonunu oluştur.

Adım 3: Her bir kromozom için uygunluk hesapla.

Eğer durdurma şartları gerçekleşmişse Adım 8'e;

Eğer durdurma şartları gerçekleşmemişse Adım 4'e git.

Adım 4: Uygunluğuna göre iki ata birey seç ve seçilen bireylerin çaprazla.

Adım 5: Seçilen bireylere belli oranlarlar mutasyon yap.

Adım6: Yeni oluşan bireyleri topluma ekle. Eski ata bireyleri popülâsyondan ele.

Adım 7: Algoritmayı sonlandır. Sonuçları göster.

Çözüm aralığı 0 ile 255 arasında 256 farklı değer alabilen bir problem olsun. Bu her bir bireyin 8 bit ile kodlanacağı anlamına gelir. Bu algoritma tek bir değeri optimize etmeye çalışır. Bunu gerçekleştiren algoritma sırayla şu adımları yapar:

1) Bireyler ikili kodda kodlanır : 1 0 0 1 1 0 0 1,0 1 1 0 0 1 1 1 veya 0 1 0 0 1 0 0 1 gibi. Aslında farklı problemler için birey daha fazla bit ile de ifade edilebilir.
2) Rasgele bireyler ile başlangıç popülasyonunu oluştur(Şekil 4.2): Popülasyondaki birey sayısı kullanıcının isteğine bırakılmıştır. Bu örnekte popülâsyondaki birey sayısı 50'dir. Ayrıca bu ilk popülasyon *popülasyon 0* olarak adlandırılır.

	2'lik sayı değeri							
Birey 1	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
Birey 2	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**
Birey 3	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**
Birey 4	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
Birey 5	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**
Birey 6	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
Birey 7	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**
Birey 8	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**
. . .	. . .					. . .	. . .	. . .
Birey 48	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**
Birey 49	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
Birey 50	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**
popülâsyon 0								

2'lik sistemde kodlanmış bireyler

Şekil 4.2: Rasgele bireyler seçerek başlangıç popülasyonu oluşturulması.

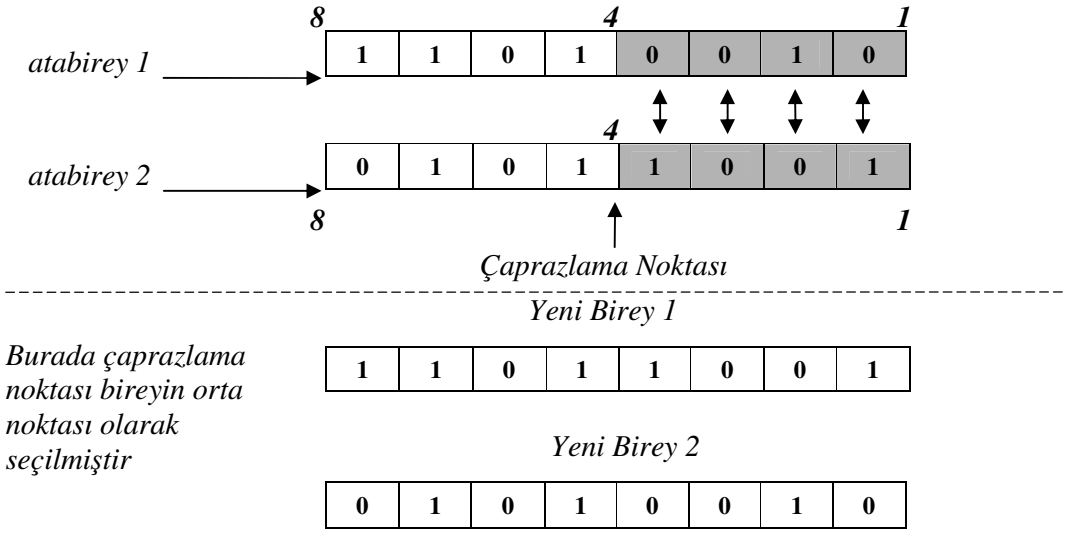
3) Bireylerin uygunluklarını ayrı ayrı hesapla (Şekil 4.3): Bu problem için birey uygunluğu bulunurken bireyin 10'luk sayı sistemindeki karşılığı hesaplanır. Daha sonra bu bir uygunluk fonksiyonundan geçirilir. Eğer birey yeterince uygun ise başarılı sayılır. Bu örnek için uygunluk fonksiyonu f(x)=2*x+5'dir. Uygunluk kriteri ise f(x)>=155'dir. Daha farklı problemlerde bu karmaşık bir fonksiyon da olabilir. Örneğin 00001010 bireyinin uygunluğu hesaplanırken ilk önce bunun 10'luk sayı sistemindeki 10 değeri bulunur. Bu sayının uygunluğu hesaplanır: f(10)=2*10+5 → f(10)=25. Sonra bu değerin uygun olup olamadığı test edilir: f(10) <255 olduğunda birey uygun değildir. Bu adım her birey için yapılır.

	2'lik sayı değeri								10'luk Değeri	f(x)	Uygun mu ?
Birey 1	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**216**	**437**	**E**
Birey 2	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**56**	**117**	**H**
Birey 3	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**97**	**199**	**E**
Birey 4	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**198**	**401**	**E**
Birey 5	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**94**	**193**	**E**
Birey 6	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**88**	**181**	**E**
Birey 7	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**14**	**33**	**H**
Birey 8	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**73**	**151**	**H**
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .			
Birey 48	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**39**	**83**	**H**
Birey 49	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**213**	**431**	**E**
Birey 50	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**84**	**183**	**E**
popülasyon 0									**toplam**		**15**

Şekil 4.3: Bireylerin uygunluklarının hesaplanması.

Durdurma şartı olan 35 uygun bireye ulaşılmadığı için algoritma 4. adıma geçer. Eğer bu sayıya ulaşılsaydı 7. adıma gidilecekti.

4) Rasgele iki ata birey seç ve çaprazla(Şekil 4.4): Bu adımda iki ata birey seçilir daha sonra bunlar bir çaprazlama noktasından çaprazlanır. Bazı problemlerde çoklu çaprazlama noktası olabilir. Bu örnek için çaprazlama noktası 4. bittir. Her iki bireyin 1 ile 4. bitleri arası karşılıklı olarak değiştirilir.



Burada çaprazlama noktası bireyin orta noktası olarak seçilmiştir

Şekil 4.4: Çaprazlama işlemi.

5) Bireyler mutasyon işlemine tabi tutulur (Şekil 4.5):

2'lik sayı değeri							
1	**0**	**1**	**1**	**1**	**1**	**0**	**1**
0	**0**	**1**	**1**	**0**	**0**	**1**	**0**
0	**1**	**1**	**0**	**1**	**0**	**0**	**1**
1	**1**	**0**	**0**	**0**	**1**	**1**	**0**
0	**0**	**0**	**1**	**1**	**1**	**1**	**0**
0	**1**	**0**	**1**	**1**	**1**	**0**	**1**
0	**0**	**0**	**0**	**1**	**0**	**1**	**0**
0	**0**	**0**	**0**	**1**	**0**	**1**	**1**
. . .					. . .	. . .	. . .
0	**0**	**1**	**0**	**0**	**0**	**1**	**0**
1	**1**	**0**	**1**	**1**	**1**	**1**	**1**
0	**1**	**0**	**1**	**0**	**1**	**0**	**0**

Mutasyondan sonra bireylerdeki değişim.

Mutasyon Noktası Seçimi:Random(9) sonuçta 1 ile 8 arasında bir nokta seçilir. Seçilen noktadaki bit değeri tersi ile değiştirilir.

Mutasyon Olasılığı:0.25 → %25
Random(100) >=25 ise değeri değiştir
Random(100) <25 ise değeri değiştirme

Şekil 4.5: Mutasyon işlemi.

6) Yeni bireyler popülasyona eklenir (Şekil 4.6):

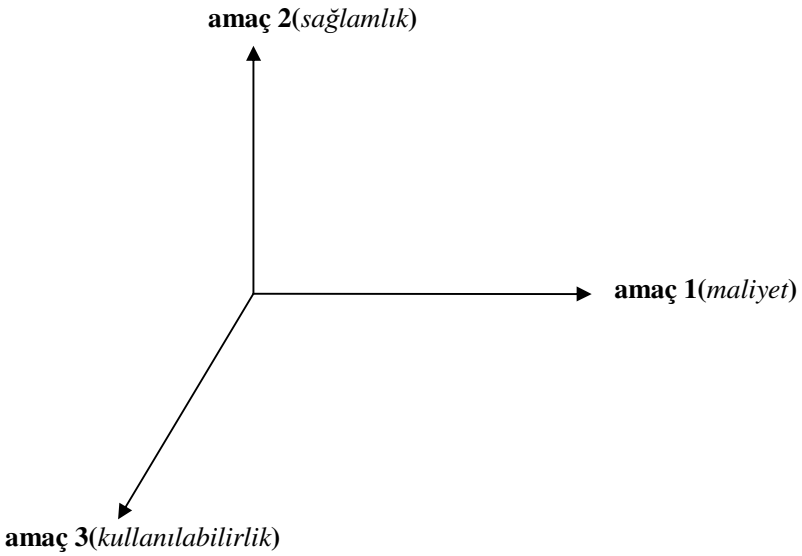
	2'lik sayı değeri							
Birey 1	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**1**
Birey 2	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**
Birey 3	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**
Birey 4	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
Birey 5	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**
Birey 6	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**1**
Birey 7	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**
Birey 8	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**
. . .	. . .					. . .	. . .	. . .
Birey 48	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**
Birey 49	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**1**
Birey 50	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**
popülasyon 1								

Şekil 4.6: Yeni bireylerin popülasyona eklenmesi.

7) Sonuçta başarılı olan bireyler kullanıcıya gösterilir.

4.2. Çok-Amaçlı Genetik Algoritma

Temel genetik algoritma tek bir sonucu optimize etmeye çalışır. Tek bir parametre için kodlama yapılır. Bazı problemlerin sonucuna tek bir parametre değil de birden fazla parametre etki eder. Bu durumda tek bir parametre ile genetik hesaplama yapmak tercih edilmez. Her bir çıkış değeri optimize edilmek istenir.



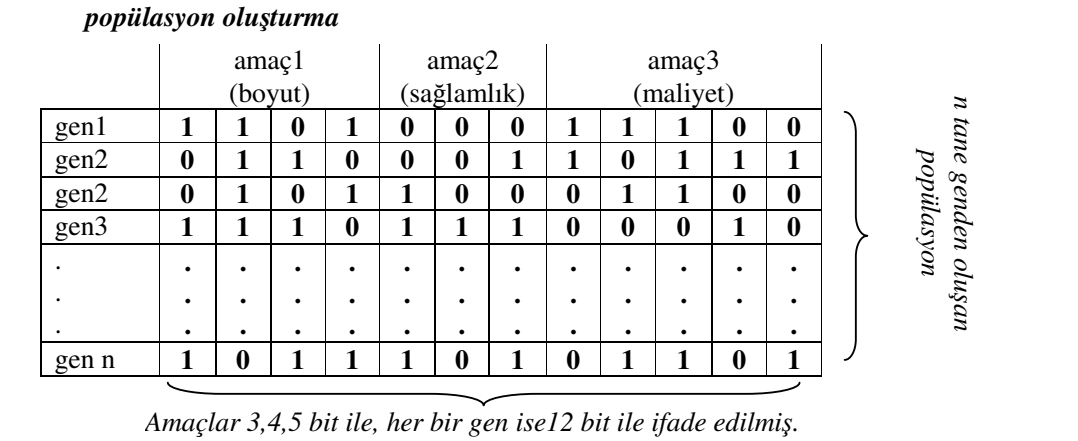
Şekil 4.7: Birden fazla amaca sahip bir problem.

Şekil4.7'de bir IPod üretimindeki birkaç etken gösterilmiştir. İlkel üretim teknolojilerinde sadece maliyet önemli iken günümüz üretim teknolojileri, üretim sürecinde olabildiğince etken kullanarak hesaplama yapar. Burada her bir parametre birbirine bağlıdır. Birine önem vermek diğerlerinden taviz vermek anlamına gelir. Örneğin sağlam bir cihaz yapmak maliyeti arttırır bunun yanı sıra kalitesiz materyalden yapılmış bir cihaz ucuz fakat kullanışsız olur. Dolayısıyla profesyonel bir üretim için parametreler optimal değerlere sahip olmalıdır.

Çok amaçlı genetik algoritma temel genetik algoritmadan geliştirilmiş bir optimizasyon tekniğidir [60]. Bu tür algoritmalarda birden fazla parametre ile işlem yapılır. Çok amaçlı genetik algoritma hedef parametrenin birden fazla olduğu evrimsel hesaplama problemlerinde de kullanır. Ürün değerlendirmesinde birden fazla uygunluk fonksiyonu ve birden fazla eşik değeri kullanılır.

Şekil 4.8'de çok amaçlı genetik algoritma kullanarak oluşturulan bir başlangıç topluluğu gösterilmiştir. Algoritma başlangıcında tüm amaçlar gen içerisinde kodlanır. Amaçlar farklı boyutlarda kodlanabileceği gibi benzer uzunlukta da kodlanabilir. Genler benzer olarak ikişer ikişer çaprazlanır ve yeni bireyler elde edilir. Mutasyon tüm gen üzerinde yapılır. Uygunluk hesabı yapılırken gen tekrar parçalara bölünür ve her amaç tek başına değerlendirilir. Her amacın uygunluk eşiği farklı değer aralığında bulunabilir.

Algoritma sonuçta tek bir iyi ürün bulmaz. İyi veya daha çok iyi sonuçların bulunduğu bir çözüm kümesi sunar. Bu da kullanıcıya daha esnek bir seçim hakkı tanır.



Şekil 4.8: Birden fazla amaç için popülasyon tanımlama.

4.3. Çok Amaçlı Genetik Algoritma ile Çoklu Motiflerin Keşfi

Daha önceki bölümlerde bahsedildiği gibi genetik alanında motif bulmak için çeşitli yöntemler geliştirilmiştir. Bunların en sık kullanılanları MEME [43] ve AlignACE'dır [47]. Bu iki algoritma için geliştirilen uygulamalar web üzerinden de yayınlanmaktadır. Gibbs Sampler dizi içerisinde en az bir tane motif olduğunu varsayar [45]. Dolayısıyla dizi içerisinden rasgele bir noktadan bir örnek alıp bunun var olup olmadığını test eder.

4.3.1. Geliştirilen Yöntem

Bu tez de öne sürülen yöntem çok amaçlı genetik algoritma kullanarak DNA üzerindeki gizli motif veya örüntüleri çıkarmaya çalışmaktadır. Algoritma birden fazla dizi üzerinde çoklu motifler bulmayı hedefler. Daha önceki yöntemler sabit uzunlukta yada belirli bir uzunluktaki motifleri ararken bu yöntem motif sayısını ve motif uzunluğunu optimal yapmaya çalışır. Daha doğrusu algoritma uzunluğu ve sayıyı kendi bulmaya çalışır.

Algoritma gerçek DNA dizileri üzerinde çalıştırılarak sonuçları analiz edilmiştir. Her birinin uzunluğu 2000 olan 4 dizi üzerinde motif araması yapılmıştır. Motif uzunlukları genel araştırmalarda 12'dir, bu yöntemde 7 ile 14 birim uzunluğundaki motifler bulunmaya çalışılmıştır.

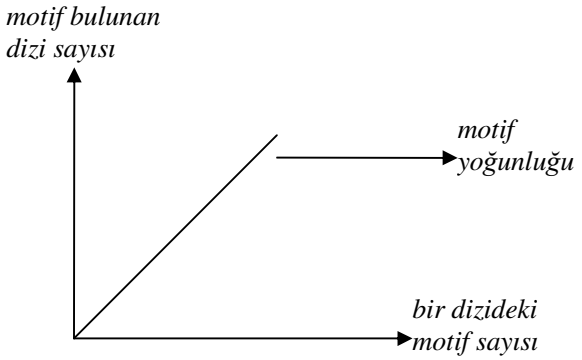
Bu algoritma 3 kriteri optimize etmeye çalışır:

1.Motif Uzunluğu: Kısa motiflerin diziler içerisinde bulunma olasılığı çok yüksektir. Fakat kısa motiflerin anlamlı veya değerli genler olması pek beklenmez. Uzun motiflerin ise aranması sırasında yapılan karşılaştırmaların çokluğu nedeniyle algoritma maliyeti artar. Bu da pek tercih edilmez. Dolayısıyla motif uzunluğu gerçek DNA üzerinde sıklıkla 12 birimdir. Bu yöntem ise motif uzunluğunu 7 ile 14 arasında optimize etmeye çalışır.

2.Motif Benzerliği: Varsayılan motif ile gizli-gerçek motifler arasındaki benzerlik motif benzerliği olarak adlandırılır. Motif benzerliği ne kadar yüksek ise motif o kadar gerçeğe yakındır ve bir motif olma olasılığı o kadar fazladır. Dolayısıyla algoritma motif benzerliğini artırmaya çalışır. Benzerlik hesaplanırken bir miktar uyumsuzluk göz ardı edilir. Benzerlik hesaplama fonksiyonu logaritmiktir.

3.Motif Yoğunluğu: Motif yoğunluğu diziler içerisinde bulunan motif sayısıdır. Algoritma motif ararken hem dizi başına birden fazla gen bulmayı hedefler hem de birden fazla dizi içerisinde aynı geni bulmaya çalışır. Bahsedilen durum Şekil 4.9'de gösterilmiştir. Bu yaklaşımda bir dizi için 0 ile 3 arasıda motif sayısı önerilmiştir. Ayrıca 4 dizi de bir motife ait

parçalar olduğu var sayılır. Dolayısıyla bir motife ait toplam maksimum 12 gen parçası bulunabilir.



Şekil 4.9: Motif yoğunluğu.

Bu yöntemde aynı miktar motifin birden fazla dizide bulunması daha fazla tercih edilir. Bu şu şekilde açıklanabilir: 1. durumda iki dizide dizi başına 3 motif olmak üzere toplam 6 motif olsun. 2. durumda ise 3 dizi de dizi başına 2 motif olmak üzere toplam yine 6 motif olsun. Burada 2. durum daha fazla kabul görür çünkü 2. durumda diziler arası benzerlik daha fazladır bu da motifin geçerliliğini artırır. Bu durum Şekil 4.10'de gösterilmiştir.

1. durum→toplam motif sayısı:6, bulunduğu dizi sayısı:2

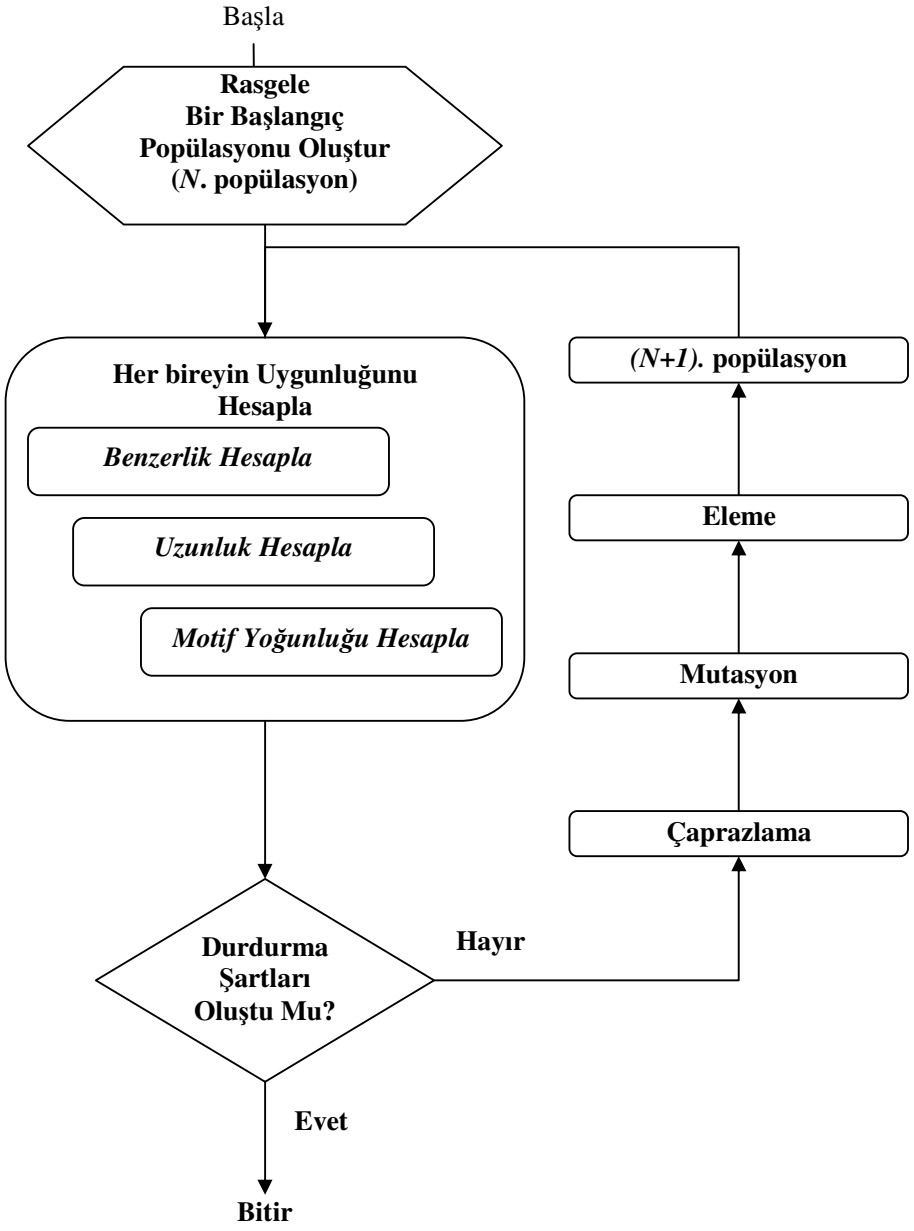
2. durum→toplam motif sayısı:6, bulunduğu dizi sayısı:3

2. durum daha değerlidir. Çünkü motif sayısı aynı olmasına rağmen 2. durumda motif bulunan dizi sayısı daha fazladır. Bu da motifin gerçeğe yakın olmasına işaret eder. Algoritma aynı miktar geni değerlendirirken kaç dizide bulunduğuna bakar.

Şekil 4.10: Gen sayısı aynı olan motiflerin değerlendirilmesi.

4.3.2. Algoritma Yapısı

Bu yöntem genetik algoritmanın tüm adımlarını içerir. Bunun yanı sıra birden fazla kriteri optimize ettiğinden uygunluk fonksiyonu birden fazla parametre değerlendirme yapar. Dolayısıyla paralel olarak çalışan 3 uygunluk fonksiyonu vardır. Önerilen yöntemin temel yapısı Şekil 4.11'de verilmiştir.



Şekil 4.11: Geliştirilen Çok-Amaçlı Genetik Algoritmanın akış diyagramı.

4.3.3. Algoritma İşleyişi

Algoritma işleyişindeki temel adımlar Şekil 4.12'de verilmiştir. Algoritma için tanımlı değerler şunlardır:

i iterasyon sayısı: Algoritmanın kaç defa çalıştırılacağını belirler.

ℓ motif uzunluğu: Bu değer algoritma tarafından optimize edilir. 7 ile 14 arasında bir değer alabilir.

∂ benzerlik değeri. Bu değeri algoritma kendisi bulur. Kullanıcı, minimum benzerlik değerini kendisi de belirleyebilir.

μ motif yoğunluğu: Dizilerde bulunan motif sayısı ile elde edilen bir değer.

$S_1..S_n$ DNA dizileri.

$m_1..m_n$ motif numaraları

Algoritma Başlangıcı:
Bireyleri kodla
İterasyon sayısını belirle.(i)
Popülasyon boyutunu belirle
Rasgele değerler kullanarak başlangıç popülasyonunu oluştur.

Algoritma:
For i = 1 **to** i (*iterasyon sayısı*) **do**
Begin

For j =1 **to** popülasyondaki_birey_sayısı **do**
Begin

motif_uzunluğunu_hesapla(*l*)
motif_ağırlıklarını_hesapla(∂)
motif_oluştur($S_1..S_n$, $m_1..m_n$)
benzerlik_hesapla()
motif_yoğunluğu_hesapla(μ)

Birey değerlendirme işlemi()

End;

Çaprazlama()
Mutasyon()
Birey değerlendirme işlemi()
Eleme()
Toplumu_Yenile()
End;

Başarılı_Bireyleri_Bul()
Birey_Bilgilerini_Göster()

Şekil 4.12: Algoritmanın işleyişi.

4.3.4. Algoritmanın Çalışması

a) Bireylerin Yapısı:

Birey (motif) yapısı Şekil 4.13'de gösterilmiştir.

	3 bit			36 bit													
	uzunluk			1. dizi için ağırlıklar			2. dizi için ağırlıklar			3. dizi için ağırlıklar			4. dizi için ağırlıklar				
				w1	*w2*	*w3*	*w4*	*w5*	*w6*	*w7*	*w8*	*w9*	*w10*	*w11*	*w12*		
birey	1	2	3	1.DİZİ			2.DİZİ			3.DİZİ			4.DİZİ				
kodu	1	0	0	1	0	0	1	----------			1	1	0	1	0	1	0	1

Bireylerin kodlanması																			
	uzunluk			w1			w2			---	w10			w11			w12		
bit no	1	2	3	4	5	6	7	8	9	---	31	32	33	34	35	36	37	38	39
birey	1	0	1	1	1	0	1	0	0	---	1	1	0	0	0	1	1	1	1

Şekil 4.13: Motif yapısı.

Bir motif parçasının bir dizi içerisinde olup olmayacağı *w*(*ağırlık*) değeri ile belirlenir. Bu yöntem için w>2 için motif dizi içerisinde olduğu varsayılır ve rasgele bir pozisyon seçilerek motif parçası bulunur. Bu şekilde her bir dizi içerisinde en fazla 3 tane motif parçası aranır. Daha sonra bir sonraki dizi içinde ağırlık değerlerine göre motif parçaları rasgele noktalar seçilerek bulunur. Bu işlem her 4 dizi için yapılır.

Motif üzerinde 12 adet ağırlık bulunur. Her ağırlık 3 bit ile ifade edilir. Dolayısıyla her ağırlık 0 ile 7 arasında değer alır. Motifin ilk 3 bit motif uzunluğu için kullanılır uzunluk 0 ile 7 arasında bir değer alır. Uzunluk 7 ile 14 arasında değiştiği için uzunluk değerine 7 eklenir ve gerçek motif uzunluğu bulunur. Tablo 4.1'de rasgele oluşturulmuş bazı ağırlık değerleri için motifin dizide olup olmadığı ve sayısı gösterilmiştir.

Tablo 4.1: Motif sayıları tablosu.

		w1	w2	w3	toplam motif sayısı
dizi1	2'lik	011	110	010	2
	10'luk	3	6	2	
	var/yok	var	var	yok	
dizi2	2'lik	111	001	010	1
	10'luk	7	1	2	
	var/yok	var	yok	yok	
dizi3	2'lik	100	000	001	1
	10'luk	4	0	1	
	var/yok	var	yok	yok	
dizi4	2'lik	001	110	110	2
	10'luk	1	6	6	
	var/yok	yok	var	var	
toplam motif sayısı					6

Tablo 4.2'de 39 bit uzunluğunda rasgele 2'lik değerden meydana gelen bireyler ile oluşturulmuş bir popülâsyon gösterilmiştir.

Tablo 4.2: 39 bit uzunluğunda rasgele oluşturulmuş popülâsyon.

birey no	**birey kodu**																				
1	1	0	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0	0	1	0	1
2	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	1
3	1	1	1	1	0	0	1	1	1	0	0	1	0	0	1	1	0	1	1	0	1
4	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	0	0	0	1
5	0	0	0	0	0	1	1	1	1	0	0	1	0	1	1	1	0	1	1	0	1
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
n	1	0	0	1	0	0	1	1	0	0	1	0	1	1	1	0	0	1	1	1	0
popülâsyon 0																					

b) Motif Oluşturma:

Motif oluşturulurken ilk önce bireyin ilk 3 biti okunarak motif uzunluğu tespit edilir. Daha sonra birinci dizide motif olup olmadığını öğrenmek için sonraki 9 bit sırayla üçer üçer okunur ve ağırlık değerleri bulunur. Eğer ağırlık 2'den büyükse 1. dizi içerisinde ilk motif rasgele bir pozisyon seçilerek motif uzunluğu kadar okunur. Bu işlem şu şekilde gerçekleşir:

1)Motif uzunluğu(ilk 3 bit): 011 olsun, 011→3'tür. Bu değere 7 eklenir ve motif uzunluğu 10 bulunur. Bundan sonra tüm motifler on uzunlukta olacaktır.

2)*w1* ağırlığı(4,5,6. bitler) 011 olsun, 011→3'tür. Bu değer 2 den büyük olduğu için 1. dizi içerisinde rasgele bir nokta seçilir ve seçilen noktadan itibaren 10 tane karakter alınır. Nokta dikkatli seçilmelidir çünkü motif dizi dışına taşabilir. 200 noktasının seçildiği kabul edilirse, bu noktadan itibaren 10 karakter şu şekildedir: ATTACGGACT ilk motif parçasıdır.

3) 2. adım 1. dizideki motif parçaları bulunana kadar devam eder. Daha sonra birey okunmaya devam ederek 2.,3. ve 4. diziler için tekrarlanır.

4) Sonuçta ilk birey için diziler içerisinden seçilen parçalardan oluşan bir motif elde edilir (Tablo 4.3):

Tablo 4.3: Motif parçalarından motif oluşturma.

1. BİREY İÇİN MOTİF TANIMLAMA										
MOTİF PARÇALARI	DNA PARÇASI									
MOTİF 1	A	T	T	A	C	G	G	A	C	T
MOTİF 2	A	T	T	A	G	G	G	A	A	A
MOTİF 3	C	C	C	T	C	G	C	T	C	T
MOTİF 4	A	A	A	T	G	C	A	A	G	G
MOTİF 5	T	G	T	A	C	A	G	A	G	G
MOTİF 6	G	G	G	A	T	T	T	G	C	A

Sonuçta 1. birey için 6 motif bulunmuştur. Bu motif daha sonraki adımlarda uygunluk fonksiyonundan geçirilir.

c) Motif Benzerliğini Hesaplama

Motif benzerliği hesaplanırken motif örneğinden yararlanarak ilk önce ağırlık matrisi oluşturulur. Şekil 4.14'de bir motif örneğinden ağırlık matrisi oluşturma gösterilmiştir. Burada motif benzerliği hesaplanırken $I_{align} = \sum_{i=1}^{l} \sum_{b \in \Sigma} f_{b,i} \log_k \frac{f_{b,i}}{p_b}$ formülü kullanılır. Burada k motif uzunluğu,$\sum$ alfabe(A,T,G,C) , p_b bazın orada olma olasılığı, $f_{b,i}$ bazın o sütundaki frekansı olarak verilir.

	1. MOTİF									
	DNA PARÇASI									
	A	T	T	A	C	G	G	A	C	T
	A	T	T	A	G	G	G	A	A	A
	C	C	C	T	C	G	C	T	C	T
	A	A	A	T	G	C	A	A	G	G
	G	G	G	A	T	T	T	G	C	A
A	3/5	1/5	1/5	3/5	0	0	1/5	3/5	1/5	2/5
T	0	2/5	2/5	2/5	1/5	1/5	1/5	1/5	0	2/5
G	1/5	1/5	1/5	0	3/5	3/5	2/5	1/5	1/5	1/5
C	1/5	1/5	1/5	0	1/5	1/5	1/5	0	3/5	0

Baz frekansları

P_b=1/5

I_1=3/5$\log_{10}$(3/5)/(1/5) + 0 +1/5*$\log_{10}$(1/5)/(1/5)+1/5*$\log_{10}$(1/5)/(1/5) →1.Sütun için benzerlik değeri*
I_2=1/5$\log_{10}$(1/5)/(1/5) + 2/5*$\log_{10}$(2/5)/(1/5) +1/5*$\log_{10}$(1/5)/(1/5) +1/5*$\log_{10}$(1/5)/(1/5) →2.Sütun için benzerlik değeri*
.
.
I_4=3/5$\log_{10}$(3/5)/(1/5) + 0 +0+2/5*$\log_{10}$(2/5)/(1/5) 4.Sütun için benzerlik değeri*
.
.
I_7=1/5$\log_{10}$(1/5)/(1/5) + 1/5*$\log_{10}$(1/5)/(1/5) +21/5*$\log_{10}$(2/5)/(1/5) +1/5*$\log_{10}$(1/5)/(1/5) →7.Sütun için benzerlik değeri*
.
.

I_{10}=2/5$\log_{10}$(2/5)/(1/5) +2/5*$\log_{10}$(2/5)/(1/5)+1/5*$\log_{10}$(1/5)/(1/5)+0 →10.Sütun için benzerlik değeri*

Sonuçta tüm sütun hizalama değerleri toplanır:

$I_{toplam}=I_1+I_2+I_3+I_4+I_5+I_6+I_7+I_8+I_9+I_{10}$

Şekil 4.14: Benzerlik değerinin hesaplanması.

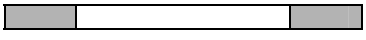
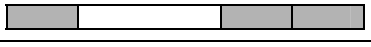
d) Motif Yoğunluğu Hesaplama

Motif yoğunluğu elde edilen motiflerin sayısı ve motiflerin kaç dizide bulunduğunu gösteren parametredir. Algoritma dizi başına birden fazla motif bulmayı amaçlar aynı zamanda bulunan motiflerin birden fazla dizide yer almasını bekler. Bu iki amaç birleştirilerek motif yoğunluğu kavramı oluşturulmuştur.

Eğer motif sayısı aynı olan bir çözüm ile karşılaşılmış ise bu durumda motiflerin kaç dizide bulunduğuna bakılır. En fazla dizi sayısına sahip olan motif tercih edilir. Çünkü motifler birbirinden bağımsız ne kadar dizide bulunursa motif o kadar gerçeğe yakın ve güçlüdür. Örneğin 1. çözümde motifler 2 dizide 3'er tane olmak üzere toplam 6 motif olsun, 2. çözümde motifler 3 dizide 2'şer adet olmak üzere toplam 6 tane olsun. Bu yaklaşıma göre 2. çözüm daha önemlidir, çünkü aynı miktar motif daha fazla dizi içerisinde bulunmaktadır.

Motif yoğunluğu şu formül ile hesaplanır:

$\mu = \sum_{n=1}^{k} \frac{m_n}{s} \sqrt{(m_n)^2 + n^2}, m_n \neq 0$ formülü kullanılarak hesaplanır. Burada m_n dizilerde bulunan motif sayıları, n ;m_n sayısının kaç dizide bulunduğunu; s dizi sayısını gösterir. Şekil 4.15'de motif yoğunluğunun hesaplanması için bir örnek verilmiştir.

	Motifler	Bulunan motif sayısı
dizi1		3
dizi2		2
dizi3		3
dizi4		2

Dizilerde bulunan motifler

m_1 →	Birli motif sayısı (Kaç dizide sadece 1 motif var?) →	0
m_2 →	İkili motif sayısı (Kaç dizide sadece 2 motif var?) →	2
m_3 →	Üçlü motif sayısı (Kaç dizide sadece 3 motif var?) →	2

s = 4 dizi var bu örnek için n=3 tür. Bu, bir dizide en fazla 3 motif bulunabilir anlamına gelir.

μ_1=0 → 1'li motif bulunmamaktadır
μ_2=2/4 * $\sqrt{(2^2 + 2^2)}$ → $\mu_2 \approx 1.4$
μ_3=2/4 * $\sqrt{(3^2 + 2^2)}$ → $\mu_2 \approx 1.8$

$\mu_t = \mu_1 + \mu_2 + \mu_3$
μ_t=0 + 1.4 + 1.8
μ_t=3.2

Şekil 4.15: Motif yoğunluğu hesaplama.

e)Çaprazlama

Çaprazlama işleminde rasgele iki ata birey seçilir. Ata bireyler 3 parçaya ayrılır. Parçalar karşılıklı olarak birbiriyle değiştirilir. Bireylerin çaprazlanması Şekil 4.16'da gösterilmiştir. Çaprazlama işlemi şu adımlardan oluşur:

Adım1: *Rasgele iki birey seç*

Adım2: *Bireyleri 3 parçaya böl.*

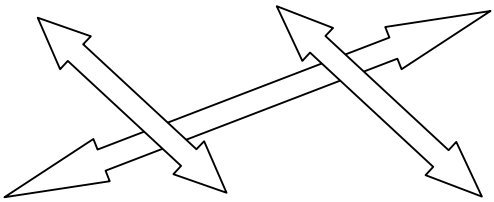
Adım3: *Parçaları karşılıklı değiştir.*

Adım 4: *Oluşan yeni bireyleri popülasyona ekle*

Adım 5: *Eğer tüm bireyler çaprazlandıysa çaprazlamayı bitir. Çaprazlanma bitmediyse Adım1'e git.*

Adım 6: *Bitir.*

atabirey1	parça1	parça 2	parça3
bit no	1←→13	14←→26	27←→39



atabirey2	parça4	parça 5	Parça6
bit no	1←→13	14←→26	27←→39

çocuk 1	parça5	parça 6	parça4
çocuk 2	parça3	parça 1	parça2

Çaprazlanacak bireyler 13'er bit olmak üzere 3 parçaya bölünür. Bu gen parçaları karşılıklı olarak ata bireyler arasında değiştirilir. Sonuçta iki yeni çocuk birey oluşur.

Şekil 4.16: Çaprazlama işlemi.

Çaprazlama işleminden sonra oluşan yeni bireyler topluma eklenir. Eski ata bireyler ise toplumdan silinmiş olur. Bundan başka sadece tek parçalı çaprazlamada yapılabilir. Bu işlem bireylerin orta noktasında bölünüp parçaların takas edilmesiyle yapılır.

f) Mutasyon

Mutasyon genetik algoritma adımlarından bir tanesidir. Her birey belli oranlarla mutasyona uğrar. Algoritma çalıştırılmadan önce mutasyon oranı belirlenir. Mutasyon yapılırken 0 ile 100 arasında rasgele bir değer üretilir. Eğer bu değer mutasyon olasılığından büyükse birey içerisinde rasgele bir nokta seçilerek o noktadaki değerin tam tersi alınır. Bu yöntemde tek bir noktaya mutasyon yapılır. Birey içerisinde birden fazla noktaya mutasyon uygulanabilir. Şekil 4.17'de bireylere mutasyon yapma işlemi gösterilmiştir.

P_m → mutasyon olasılığı 0.05 olsun,
Eğer birey mutasyon olasılığı 95 ve üzeri bir değer alırsa
bireye mutasyon yapılır. Aksi halde mutasyon yapılmaz.

P_1=Random(100) → p_1=96 olsun
1.Bireye mutasyon yapılır. (P_1>=P_m)

$m_{noktası}$=Random(39) → $m_{noktası}$=4
1 ile 39 arasında rasgele bir mutasyon noktası seçilir. (Birey uzunluğu 39 bit olduğu için)
12. bit değiştirilecek

Bit no	1	2	3	4	5	6	7	8	9	10	11	12		--------		37	38	39
Bit değeri	1	0	1	0	1	1	1	0	1	0	0	1		--------		0	0	1

eski birey

Bit no	1	2	3	4	5	6	7	8	9	10	11	12		--------		37	38	39
Bit değeri	1	0	1	1	1	1	1	0	1	0	0	1		--------		0	0	1

yeni birey

Şekil 4.17: Mutasyon işlemi.

Bu tezde önerilen yaklaşım genetik algoritmanın tüm adımlarını içermektedir.

5. UYGULAMA SONUÇLARI

5.1. Algoritma Test Sonuçları

Geliştirilen yöntem gerçek DNA dizileri kullanılarak test edilmiştir. Test aşamasında; internet üzerindeki diğer motif keşfetme uygulamaları ile elde edilen sonuçlar, önerilen yöntemin sonuçları ile karşılaştırılmıştır.

5.1.1 Test Kriterleri

Bu yöntem 3 kriteri optimize etmeye çalışmıştır. Bunlar motif uzunluğu, motif benzerliği ve motif yoğunluğu. Algoritmanın uygulama programı bu 3 kriterin değişimini izler. Algoritma bu 3 kriteri kendi kendine optimize etmeyi hedefler. Dolayısıyla geliştirilen uygulamada algoritmaya uygun olarak kendi kendine analiz yapan bir sistem olarak çalışır.

Test aşamasında algoritma karmaşıklığını artırmamak için bazı kısıtlamalar da uygulanmıştır. Örneğin motif uzunluğu minimum 7 maksimum 14 olabilir. Birçok yöntem sabit 12 birim uzunluğundaki motifleri bulmayı dener. Bu yöntem ise motif uzunluğunu bulmak için daha esnek davranır.

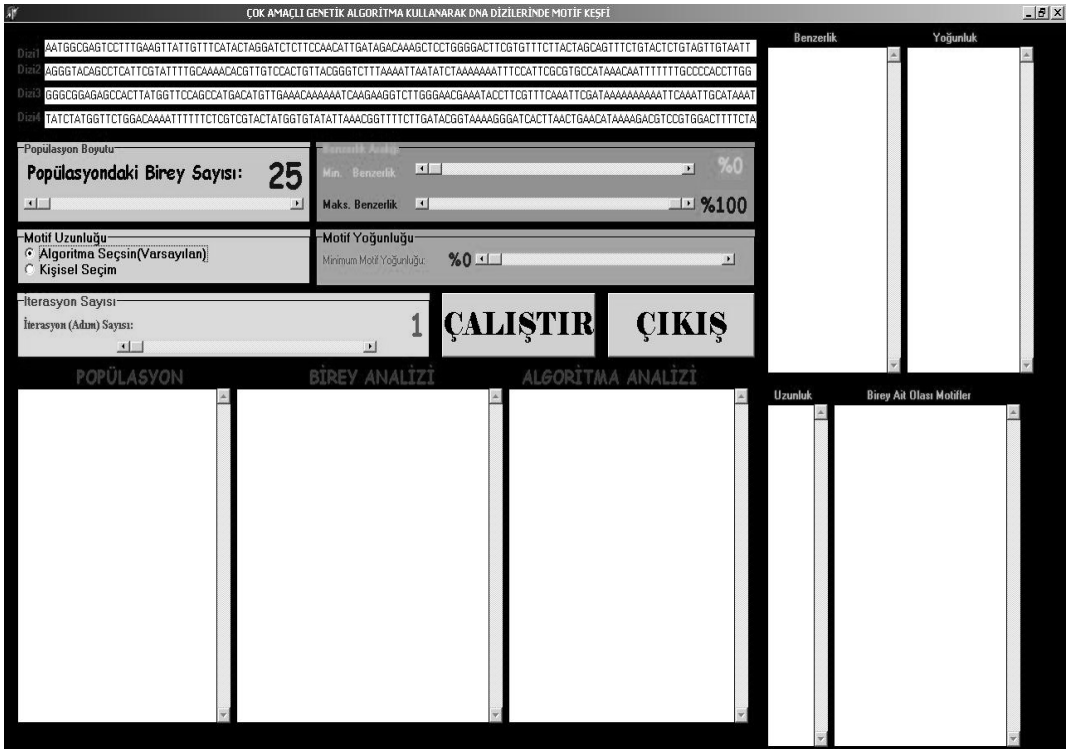
Uygulamada dizi uzunluğu 2000 olan toplam 4 dizi kullanılmıştır. Kullanıcı uygulama esnasında iterasyon sayısını ve popülasyon sayısını değiştirebilir. Popülasyon büyüklüğü standart olarak 25 alınır.

5.1.2 Test Ortamı

Testlerin yapıldığı bilgisayar şu donanımının özelliklerine sahiptir: işlemci Amd Turion 64 X2 Mobile, çift çekirdekli işlemci, işletim sistemi Windows XP Professional, RAM bellek 1024 MB. Program Delphi 6.0 Builder.

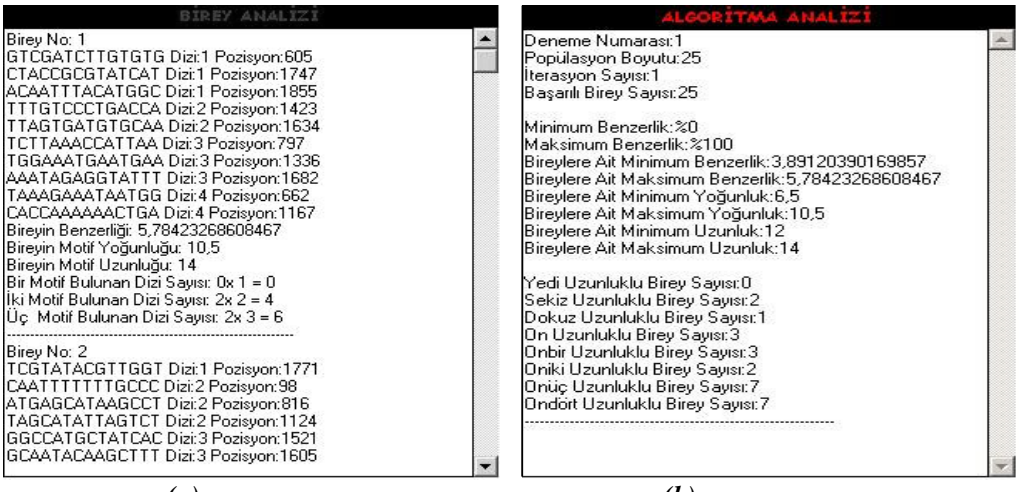
5.1.3 Geliştirilen Uygulama

Bu tezde anlatılan yöntem için hazırlanan uygulama programının kullanıcı arayüzü Şekil 5.1'de verilmiştir. Kullanıcı program çalışırken algoritmada verilen kriterlerin sınırlarını kısmi olarak değiştirebilir.



Şekil 5.1: Uygulama programının kullanıcı ara yüzü.

Program çalıştırıldığında Şekil 5.2'ye benzer sonuçlar verir. Program sonuçta başarılı bireyin motif parçaları hakkında bilgi verir. Buradan buluna motifler hakkında kısa ve öz bilgiler edinilebilir.



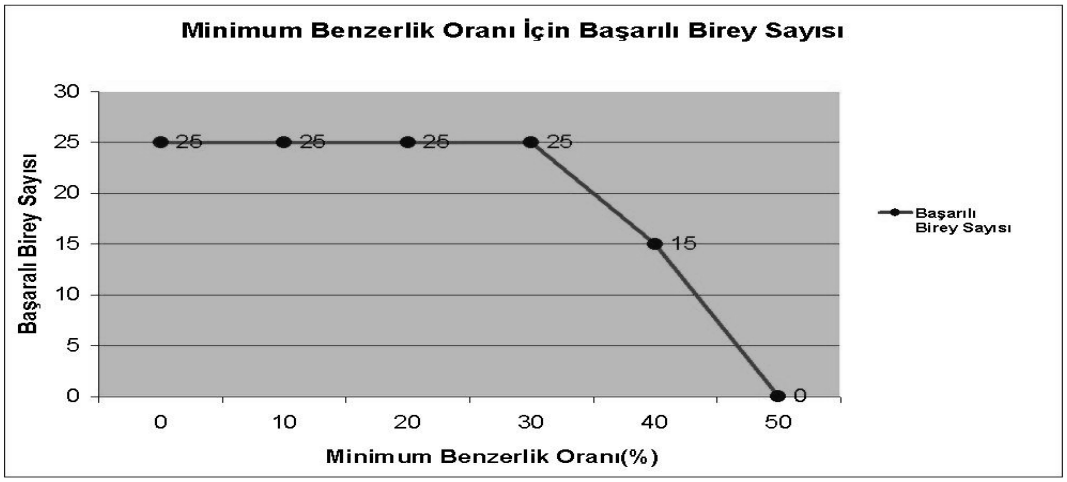
(a) *Başarılı bireye ait sonuçlar.*

(b) *Bir iterasyon sonucu.*

Şekil 5.2: Uygulama programının verdiği sonuçlar.

5.2 Elde Edilen Sonuçlar

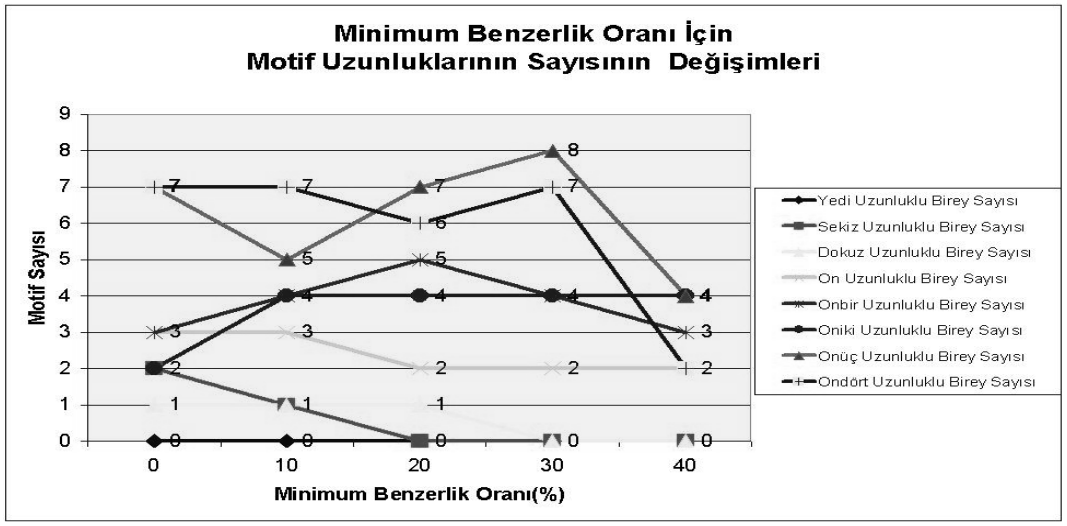
Uygulamadan elde edilen sonuçlar başarılı bireyler için verilmiştir. Burada başarı ölçüsü motifin minimum benzerlik değerinden daha fazla benzerlik değerine sahip olmasıdır. Şekil 5.3'de minimum benzerlik değerleri için başarılı birey sayısı gösterilmiştir.



Şekil 5.3: Minimum benzerlik değerini geçen birey sayısı.

Minimum benzerlik değeri arttıkça başarılı birey sayısı azalmaktadır. Düşük minimum benzerlik değerleri için uygunluğu az olan daha fazla sayıda motif bulunabilir.

Şekil 5.4'de minimum benzerlik değerinin motif uzunluğu üzerindeki etkisi görülmektedir.

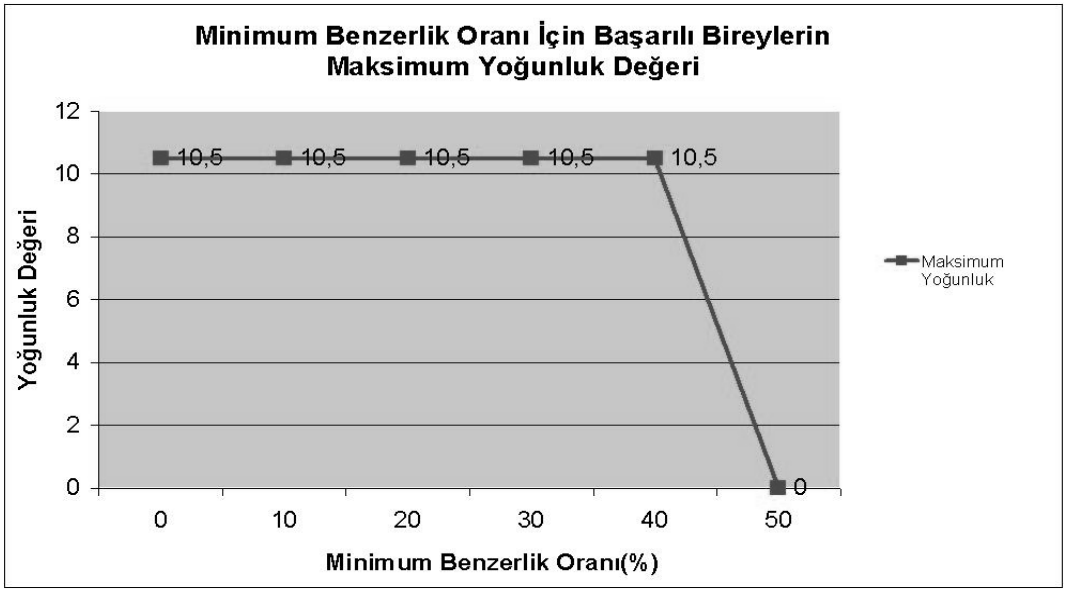


Şekil 5.4: Minimum benzerlik değeri için motif uzunlukları.

Motif uzunluğunun artması motifler arasındaki benzerliği düşürmektedir. Bu da motiflerin uyumsuzluğunun artmasında neden olur. Motif uyumsuzluğu minimum benzerlik altında kalırsa motif başarısız sayılır. Kısa uzunluk motiflerin birbirine benzer olma olasılığı daha fazla olduğu için düşük benzerlik değerlerinde bile eşik değerine geçer ve başarılı sayılırlar.

Motif benzerliği motif yoğunluyla da ilişkilidir. Motif yoğunluğu arttıkça motif sayısı da artar. Motif sayısının artması motifler arası benzerliği düşürür. Dolayısıyla motifler minimum benzerlik değerini geçemezler ve başarısız sayılırlar. Şekil 5.5'de motif yoğunluğu ile minimum benzerlik arasındaki ilişki gösterilmiştir.

Eğer yoğunluğu yüksek motifler aranmak istenirse burada minimum benzerlik değeri düşük verilmelidir. Aksi halde olası motifleri bulmak zorlaşır.



Şekil 5.5: Minimum benzerlik değeri için motif yoğunluğu grafiği.

Motif yoğunluğu en iyi 12–14 değerleri arasında optimal çözümü vermiştir. Bu da diğer yöntemler ile benzerlik gösterir. Diğer birçok yöntem uzunluğu 12 olan motiflerin daha fazla bulunabileceği konusunda benzer görüşe sahiptirler.

Yapılan test sonucunda motif aramasında benzerlik değerinin diğer tüm kriterleri etkilediği gözlenmiştir. Uzun motiflerin DNA içerisinde bulunması minimum benzerlik değerinin düşürülmesi ile olabilir. Ayıca motif yoğunluğu fazla olan bireyler daha düşük benzerlik sahibine sahiptirler. Bunun yanı sıra motif yoğunluğu motif gerçekten var olma olasılığını artırır. Farklı dizilerde bulunan motif parçaları daha fazla öneme sahiptirler. Fakat motif yoğunluğu yüksek bireyler daha az benzerlik gösterir. Bu da bireyin başarısını düşürür.

6. SONUÇLAR

Genetik alanında son yıllarda meydana gelen hızlı gelişmeler insanların büyük ilgisini çekmektedir. İnsan genomu projesi ile bilim adamları insanlık için şu anda büyük tehdit olan birçok probleme çözüm getirmeyi hedeflemektedir.

Genetik ve mikrobiyoloji alanındaki bu büyük ilerlemelere bilişim dünyasındaki gelişmeler büyük katkıda bulunmuştur. Birçok bilgisayar bilimi araştırmacısı DNA'nın sırlarını çözmek için çeşitli yöntemler geliştirmişlerdir. Genler üzerinde araştırma yapma zaman alan maliyetli bir iştir. Bu yüzden bilim adamları hızlı ve akıllı sistemleri normal çözüm yöntemlerine tercih etmektedir.

Bu tezde biyoenformatik alanında kullanılabilecek yeni bir yöntemden bahsedilmiştir. Daha önce bu alanda kullanılan yöntemler incelenerek yöntemlere farklı bir yaklaşım getirilmiştir. Bu yaklaşım genetik algoritmanın alt yapısını kullanmaktadır. Çoğu genetik algoritma tek bir amacı gerçekleştirmek için çalışırlar. Ama bazen çözüm sadece bir amaca göre şekillenmeyebilir. Bu tür durumlarda sistemi etkileyen tüm parametreler çözüm üretmek için kullanılmalıdır.

Bu tezde birden fazla parametreyi optimize etmeye çalışan bir yöntem anlatılmıştır. DNA üzerinde motif çıkaran daha önceki algoritmalar ile benzerlik gösterse de çoklu parametre ile çalışması algoritmanın kullanışlığını arttırmaktadır. Geliştirilen bu yöntemde bir çözüm havuzu sunulmaktadır. Algoritma çözümler içerisinden en iyi olanları bulmaya çalışır. Algoritmadaki her bir parametre bir diğeri üzerinde etki yapar. Bu yüzden parametrelerin değerleri itina ile seçilmelidir. Yöntem parametreleri seçerken birbirine etkilerini azaltmayı hedefler.

Algoritma kaynak olarak tek bir diziyi almaz. Birden fazla dizi üzerinde birden fazla motif bulmayı bekler. Bu yöntem çok dizi hizalamadan biraz farklıdır. Çoklu dizi hizalamada diziler bir motif grubu için sağa sola kaydırılır. Bu yöntemde dizilerin kaydırılması söz konusu değildir. Çoklu dizi hizalamada motif pozisyonları birinci önceliklidir. Çoklu dizi hizalamalarının çoğunda pozisyon genetik algoritma içerisinde bireyi tanımlayan bir parçadır. Bu tezdeki yöntemde pozisyon birey içerisinde kodlanmaz. Birey sadece motifin dizi içerisinde varlığı veya yokluğuyla ilgilenir.

Algoritmada motif benzerliği(uygunluğu) diğer iki parametreden(uzunluk ve yoğunluktan) daha baskındır. Uygunluk hesabında kullanılan yöntemin geliştirilmesi algoritmanın güvenilirliğini artıracaktır.

Algoritmanın sonucunu görmek için bir uygulama programı hazırlanmıştır. İlerleyen zaman dilimi içerisinde benzer uygulamanın web ortamına aktarılması ile araştırmacılar uzaktan veri göndererek sonuçlarını alabilecektir. Bu da biyoenformatik alanına bir parça katkıda bulunacaktır.

KAYNAKLAR

1. Ladish H., Berk A., Zipursky S., Matsudama P., Baltimore D. and Darnell J, *Molecular Cell Biology*, 2000.

2. Hames,B.D and Higgins,S.J., Gene Transcription: a practical approach. Oxford University Press Inc., NewYork, 1993.

3. Baldi P., and Hatfield G. W., "DNA Microarrays and Gene Expression: From Experiments to Data Analysis and Modeling", *Cambridge University Press*, September 2002.

4. Bourne F. E., Weissig H., "Structural Bioinformatics", John Wiley, February 2003.

5. Baldi P. and Brunak S., "Bioinformatics: The Machine Learning Approach", MIT Press, August 2001.

6. Xiong J., Essential Bioinformatics, Cambridge University Press, March 2006.

7. Hu J., Li B., Kihara D.,"Limitations and Potentials af Current Motif Discovery Algorithms", *Nucleic Acids Research*, 2005.

8. National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov/

9. Nordic Gene Bank, http://www.ngb.se/

10. Gene-regulation.com, http://www.gene-regulation.com/pub/databases.html

11. Tsai H. K., Yang J. M., Tsai Y. F., Kao C. Y.,"An Evolutionary Approach For Gene Expression Patterns", *Information Technology in Biomedicine,* 2004.

12. Shinozaki, D., Akutsu, T. , Maruyama, O., "Finding Optimal Degenerate Patterns in DNA Sequences.", *Bioinformatics,* 2003.

13. Chang B. C. H., Ratnaweera A., Halgamuge S. K., and Watson H. C., "Particle Swarm Optimisation for Protein Motif Discovery", Genetic Programming and Evolvable Machines Volume 5, Number 2 / June, 2004.

14. National Human Genome Research Institute, http://www.genome.gov/

15. Marvin E. Frazier, Gary M. Johnson, David G. Thomassen, Carl E. Oliver, Aristides Patrinos, "Realizing the Potential of the Genome Revolution: The Genomes to Life Program", *Science* 300, 290, 2003.

16. Benson S. Y. Lam, Hong Yan, "A New Similarity Measure For Microarray Data Analysis", *IEEE Transactions On Knowledge And Data Engineering,* 2005.

17. Yao Z., Weinberg Z. and Ruzzo W. L., "CMfinder—a covariance model based RNA motif finding algorithm", *Bioinformatic*s, 22(4):445-452, 2006.

18. D. GuhaThakurta, "Computational identification of transcriptional regulatory elements in DNA sequence", *Nucleic Acids Res*., 34(12): 3585 – 3598, July 19, 2006.

19. Doğruel M.**,** Down T. A. and Hubbard T. JP, "NestedMICA as an ab initio protein motif discovery tool", *BMC Bioinformatics*, 9**:**19, 2008.

20. Sapiyan M., "DNA Structure, Replication, Transcription, and Protein Synthesis", *Educational Technology & Society* 1(1) 1998.

21. Tompa M. et al., "Assessing Computational Tools for the Discovery of Transcription Factor Binding Sites", *Nature Biotechnology*, vol. 23, no. 1, 137 – 144, January 2005.

22. Chaichoompu, K. Kittitornkun, S. Tongsima, S., "MT-ClustalW: multithreading multiple sequence alignment", *20th International Parallel and Distributed Processing Syposium*, 2006.

23. Notredame, C.,Higgins, D. G., "SAGA: Sequence Alignment By Genetic Algorithm.", *Nucleic Acids Research*, s:1515-24, 1996.

24. Modan Das, Ho K Dai, "A survey of DNA motif finding algorithms", *BMC Bioinformatics*, Vol. 8, No. Suppl 7. 2007.

25. Lones, M.A.; Tyrrell, A.M, "Regulatory Motif Discovery Using a Population Clustering Evolutionary Algorithm", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, s:448 – 452, 2007.

26. F. F. M Liu, Jeffrey J.P. Tsai, R.M Chen, S.N. Chen and S.H. Shih "FMGA: Finding Motifs by Genetic Algorithm", *Proc. of BIBE'04* , s:459-466, 2004.

27. Sinha S., and Tompa M., "A statistical method for finding transcription factor binding sites", *Proc Int Conf Intell Syst Mol Biol,* **8**, 344–54, 2000.

28. D'haeseleer P. "What are DNA sequence motifs? National Biotechnology, 24, 423–425, 2006.

29. Brazma A., Jonassen I., Eidhammer I. and Gilbert D., "Approaches to the automatic discovery of patterns in biosequences", *J Comput Biol,* 5 (2), 279–305, 1998.

30. Galas, D., Eggert, M. & Waterman, M., "Rigorous patternrecognition methods for DNA sequences. Analysis of promoter sequences from Escherichia coli", *J Mol Biol,* 186 (1), 117–28, 1985.

31. Rigoutsos, I. & Floratos, A. "Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm", *Bioinformatics,* 14 (1), 55–67, 1998.

32. Bussemaker, H., Li, H. & Siggia, E. "Regulatory element detection using a probabilistic segmentation model", *Proc Int Conf Intell Syst Mol Biol,* 8, 67–74, 2000.

33. Hertz, G. & Stormo, G. "Identifying DNA and protein patterns with statistically significant alignments of multiple sequences", *Bioinformatics,* 15 (7-8), 563–77, 1999.

34. Liang, S., Samanta, M. & Biegel, B. "cWINNOWER algorithm for finding fuzzy dna motifs", *J Bioinform Comput Biol,* 2 (1), 47–60, 2004.

35. Sagot, M.-F. “Spelling approximate repeated or common motifs using a suffix tree”, In *Proceedings of the Third Latin American Symposium on Theoretical Informatics* pp. 374–390 Springer-Verlag, 1998.

36. Marsan, L. & Sagot, M. “Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification”, *J Comput Biol,* 7 (3-4), 345–62, 2000.

37. Eskin, E. & Pevzner, P. A. “Finding composite regulatory patterns in DNA sequences”, *Bioinformatics,* 18 Suppl 1, S354–63, 2002.

38. Keich, U. & Pevzner, P. “Finding motifs in the twilight zone”, *Bioinformatics,* 18 (10), 1374–81, 2002.

39. Buhler, J. & Tompa, M. “Finding motifs using random projections”, *J Comput Biol,* 9 (2), 225–42, 2002.

40. Goldberg, D. E. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley, 1989.

41. Hernandez, D., Gras, R. & Appel, R. “MoDEL: an efficient strategy for ungapped local multiple alignment”, *Comput Biol Chem,* 28 (2), 119–28, 2004.

42. Haußler M.,“Motif Discovery on Promotor Sequences”, *Master Thesis,* University of Postdam, 2005.

43. Bailey, T. L., “Discovering motifs in DNA and protein sequences:The approximate common substring problem”, *PhD thesis*, University of California, San Diego, 1995.

44. Xing, E. P., Wu, W., Jordan, M. I. & Karp, R. M. “Logos: a modular bayesian model for de novo motif detection”, *J Bioinform Comput Biol,* 2 (1), 127–54, 2004.

45. Lawrence, C., Altschul, S., Boguski, M., Liu, J., Neuwald, A. & Wootton, J. “Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment”, *Science,* 262 (5131), 208–14, 1993.

46. Neuwald, A., Liu, J. & Lawrence, C. “Gibbs motif sampling: detection of bacterial outer membrane protein repeats”, *Protein Sci,* 4 (8), 1618–32, 1995.

47. Hughes, J., Estep, P., Tavazoie, S. & Church, G. “Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae”, *J Mol Biol,* 296 (5), 1205–14, 2000.

48. Workman, C. & Stormo, G. 7048258. *Pac SympBiocomput,* 5, 467–78, 2000.

49. Thijs, G., Lescot, M., Marchal, K., Rombauts, S., Moor, B. D., , P. & Moreau, Y. “A higher-order background model improves the detection of 111 Bibliography promoter regulatory elements by Gibbs sampling”, *Bioinformatics,* 17 (12), 1113–22, 2001.

50. Liu, X., Brutlag, D. & Liu, J. "BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes", *Pac SympBiocomput,* 7, 127–38, 2001.

51. Liu, X. S., Brutlag, D. L. & Liu, J. S. "An algorithm for finding protein-DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments", *Nat Biotechnol,* 20 (8), 835–9, 2002.

52. GuhaThakurta, D. & Stormo, G. "Identifying target sites for cooperatively binding factors", *Bioinformatics,* 17 (7), 608–21, 2001.

53. http://www.cse.ucsc.edu/~kent/improbizer/index.html

54. Favorov, A., Gelfand, M., Mironov, A. & Makeev, V. "Yet another digging for dna motifs gibbs sampler", In *Proceedings of the Third International Conference on Bioinformatics of Genome Regulation and Structure, BGRS 2002,Novosibirsk, Russia* vol. 1, pp. 31–33, 2002.

55. Zhou, Q. & Liu, J. S. "Modeling within-motif dependence for transcription factor binding site predictions", *Bioinformatics,* 20 (6), 909–16, 2004.

56. Edgar R. C., "MUSCLE: a multiple sequence alignment method with reduced time and space complexity", *BMC Bioinformatics*, **5:**113, 2004.

57. Che D., Song Y., and Rasheed K.,"MDGA: Motif Discovery using A Genetic Algorithm", *Proc. of GECCO'05 s:447-452,* 2005.

58. Paul T. K., Iba H.,"Identification of Weak Motifs in Multiple Biological Sequences using Genetic Algorithm", *In Proc. of GECCO*, 2006.

59. Holland, J. H. "Adaptation in natural and artificial systems", Cambridge, MA: MIT Press, The MIT Press edition, 1992.

60. Deb,K. et al., "A fast and elitist multi-objective genetic algorithm: NSGA II", IEEE Trans. Evolutionary Computation, 6, 182-197, 2002.

ÖZGEÇMİŞ

Melikali GÜÇ

maliguc@hotmail.com

Fırat Üniversitesi
Bilgisayar Mühendisliği Bölümü
23119, Elazığ

Melikali GÜÇ, 1982 yılında Giresun'da doğdu. İlköğretimini Ordu ve Bitlis'te, ortaöğretimini de Giresun'da tamamladı. 2000 yılında Giresun Anadolu Öğretmen Lisesi'nden mezun oldu. 2000 yılında üniversite sınavına girdi ve Fırat Üniversitesi Bilgisayar Mühendisliği Bölümü'nde lisans eğitimi almayı hak kazandı. 2005 yılında mezun olup aynı yılın Eylül ayında Fırat Üniversitesi Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimi almaya başladı.