EÜ FEN BİLİMLERİ ENSTİTÜSÜ

EGE ÜNİVERSİTESİ

DOKTORA TEZİ

GENOMİK VERİTABANLARINDA İNDEKSLEME VE ARAMA YÖNTEMLERİ ÜZERİNE

Deniz TANIR

Tez Danışmanı : Prof. Dr. Urfat NURİYEV

Matematik Anabilim Dalı

Sunuş Tarihi : 27.10.2017

Bornova-İZMİR
2017

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(DOKTORA TEZİ)

GENOMİK VERİTABANLARINDA İNDEKSLEME VE ARAMA YÖNTEMLERİ ÜZERİNE

Deniz TANIR

Tez Danışmanı: Prof. Dr. Urfat NURİYEV

Matematik Anabilim Dalı

Sunuş Tarihi: 27.10.2017

Bornova – İZMİR

2017

Deniz TANIR tarafından doktora tezi olarak sunulan "Genomik Veritabanlarında İndeksleme ve Arama Yöntemleri Üzerine" başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 27.10.2017 tarihinde yapılan tez savunma sınavında aday oybirliği/~~oyçokluğu~~ ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	: Prof. Dr. Urfat NURİYEV	
Raportör Üye	: Yrd. Doç. Dr. Arif GÜRSOY	
Üye	: Doç. Dr. Burak ORDİN	
Üye	: Doç. Dr. Murat Erşen BERBERLER	
Üye	: Yrd. Doç. Dr. Hakan KUTUCU	

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Doktora Tezi olarak sunduğum "Genomik Veritabanlarında İndeksleme ve Arama Yöntemleri Üzerine" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

27 / 10 / 2017

Deniz TANIR

ÖZET

GENOMİK VERİTABANLARINDA İNDEKSLEME VE ARAMA YÖNTEMLERİ ÜZERİNE

TANIR, Deniz

Doktora Tezi, Matematik Anabilim Dalı

Tez Danışmanı: Prof. Dr. Urfat Nuriyev

Ekim 2017, 71 sayfa

Biyoenformatiğin en önemli konularından iki tanesi büyük genomik veritabanlarında gen dizilimleri arama ve genomik verilerin kümelenmesidir. Genomik veri tabanlarında milyarlarca nükleotit yer almaktadır ve bu veritabanlarında hızlı arama yapabilmek ve elde edilen sonuçları doğru analiz edebilmek araştırmacılar için çok önemlidir. Bu konuda yapılan çalışmalar verilerin etkin şekilde indekslenmesi gerekliliğini ortaya koymaktadır. Ayrıca gen dizilerinin fonksiyonel işlevlerinin belirlenmesi, genetik hastalıkların anlaşılması ve tedavi yöntemlerinin geliştirilmesi için bu verilerin kümelenmesi gen analizlerinde büyük kolaylık sağlamaktadır. Genomik verilerin kümelenmesi için de literatürde yapılan birçok çalışma mevcuttur.

Bu tez çalışmasında, genomik veriler için geliştirilen arama, indeksleme ve kümeleme yöntemleri araştırılmıştır. Genomik verilerin kümelenmesi için yeni kümeleme yöntemleri önerilmiştir. Önerilen yöntemlerin sonuçları literatürdeki diğer kümeleme algoritmaları ile karşılaştırılmıştır. Elde edilen sonuçlar önerilen yöntemin literatürdeki diğer algoritmalardan daha iyi kümeleme yaptığını göstermiştir.

Son olarak bu tez çalışmasında arama ve indeksleme teknolojilerinden yararlanan yazılım programları geliştirilmiştir.

Anahtar sözcükler: Biyoenformatik, genomik veritabanları, arama, indeksleme, kümeleme, bilgisayar yazılımı, veri madenciliği, hizalama, dinamik programlama

ABSTRACT

ON INDEXING AND SEARCHING METHODS FOR GENOMIC DATABASES

TANIR, Deniz

PhD in Mathematics
Supervisor: Prof. Dr. Urfat Nuriyev
October 2017, 71 pages

Two of the most important subjects of bioinformatics are searching gene sequences in large genomic databases and clustering genomic data. There are billions of nucleotides in genomic databases, and it is very important that researchers search the databases quickly and analyze the results accurately. Studies in this area shows that data should be efficiently indexed. In addition, clustering of these sequences provides great advantage in gene analysis which are the identification of functional sequences of gene sequences, understanding of genetic diseases and development of treatment methods. There are many studies in the literature for the clustering of genomic data.

In this thesis, searching, indexing and clustering methods have been researched. New clustering methods have been proposed. The results of the proposed methods are compared with other clustering algorithms in the literature. The results show that the proposed methods performs better than other algorithms in the literature.

Finally, in this thesis software programs have been developed using searching and indexing technologies.

Keywords: Bioinformatics, genomic databases, searching, indexing, clustering, computer software, data mining, alignment, dynamic programming

TEŞEKKÜR

Bu tez çalışmasında her daim desteğini, bilgi ve tecrübelerini benden esirgemeyen danışman hocam **Prof. Dr. Urfat NURİYEV**'e ve bioenformatik alanındaki çalışmalarımda yardımlarını esirgemeyen **Prof. Dr. Afig BERDELİ**'ye teşekkürlerimi sunarım.

Doktora çalışmalarım boyunca maddi ve manevi desteklerini hiçbir zaman esirgemeyen aileme ve tez çalışmamı bitirmemde bana güç veren arkadaşım **Uzm. Aslı Beril KARAKAŞ**'a teşekkürlerimi sunarım.

Ayrıca 2228-B doktora burs programı ile dotora eğitimim boyunca bana maddi destek veren **TÜBİTAK-BİDEB**'e ve **2016/Fen/050** numaralı proje ile tez çalışmamı destekleyen **Ege Üniversitesi Bilimsel Araştırma Proje Fonuna** teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (DEVAM)

Sayfa

İÇİNDEKİLER (DEVAM)

Sayfa

İÇİNDEKİLER (DEVAM)

Sayfa

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKİLLER DİZİNİ (DEVAM)

Şekil Sayfa

ŞEKİLLER DİZİNİ (DEVAM)

Şekil Sayfa

ÇİZELGELER DİZİNİ

Çizelge Sayfa

KISALTMALAR DİZİNİ

Kısaltma	Açıklama
NCBI	Ulusal Biyoteknoloji Bilgi Merkezi
EMBL	Avrupa Moleküler Biyoloji Laboratuvarı
DDJB	Japonya'nın DNA Veri Bankası
PDB	Protein Bilgi Bankası
CDS	Kodlama Dizisi
cDNA	Tamamlayıcı DNA
İGP	İnsan Genom Projesi

1. GİRİŞ

İnsanoğlunun en temel çabası var olduğundan beri dünya ve canlıları anlamak üzerine olmuştur. Bunların başında insanlığın ve canlıların nasıl oluştuğu ve bunu evrim teorisine bağlayan gizem gelmektedir. İnsan genom projesi ile birlikte insan genom yapısının tümünün ortaya çıkarılması amaçlandı ve proje 2003 yılında tamamlandı. Bu projeyle birlikte elde edilen veriler çok değerli olmasına rağmen bu veriyi anlamak ve veride arama yapmaya çalışmak bazı problemleri ortaya çıkardı. Öncelikle verinin boyutu çok büyük ve ayrıca içeriğinde büyük boyutta gereksiz veri barındırmaktadır. Bu durum DNA'da meydana gelen bazı mutasyonlardan kaynaklanmaktadır. Bu mutasyonların sonucunda zarar görmüş tamamlanamayan gen yapıları oluşur. Bazı durumlarda bu gereksiz veriler de türlerin nasıl geliştiği konusundaki sorulara cevap vermesi sayesinde değerli olabilmektedir. Sıklıkla veri tekrarları da karşımıza çıkmaktadır ve bu durum yararlı ve yararsız verilerin hangileri olduğunu anlamayı zorlaştırmaktadır.

Son yıllarda DNA ve onun uygulamaları çok dikkat çekmiş durumdadır. Bu uygulamalar tıpta ve ayrıca kriminal araştırmalarda karakteristiklerin ortaya çıkarılmasında etkin rol oynamaktadır. Yaşamı anlamamızı sağlayacak ve evrim bilimindeki gizemleri aydınlatacak sayısız keşfedilmeyi bekleyen DNA dizileri mevcuttur. Klonlama, genetik hastalıkların tedavisi ve kök hücre ile organ yenilenmesi gibi konularda DNA dizilerinin analiz çalışmaları yapılmaya devam etmektedir.

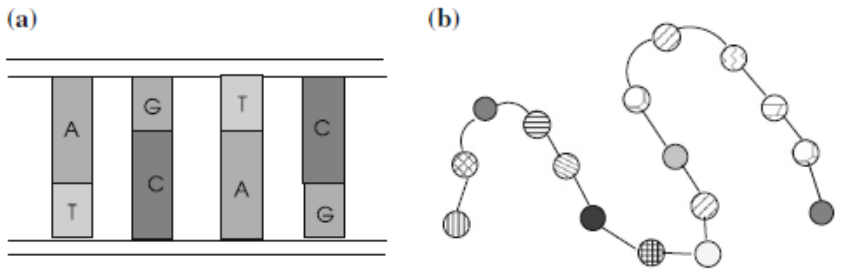
Biyoenformatik, canlıların moleküler düzeydeki verileri anlamaya, yönetmeye ve düzenlemeye çalışan matematik, istatistik, biyoloji ve bilgisayar bilimlerinin birleşiminden meydana gelen bilim dalıdır. Bu bilim dalının amacı biyolojik verilerin analizi, anlaşılması, bilgilerin organize edilerek hipotezler oluşturulmasına kolaylık sağlanmasıdır. Moleküler biyoloji ve genetik gibi alanlarda çalışan araştırmacılar için biyoenformatiğin sağladığı olanaklardan yararlanmak kaçınılmaz hale gelmiştir.

James D. Watson ve Francis Crick'in DNA'nın yapısını çözmesiyle birlikte (Watson ve Crick, 1953) ortaya çıkan DNA dizilimlerinin şifresini ortaya çıkarmak için yıllardır çalışmalar yapılmaktadır. Başta İnsan Genom Projesi (İGP) olmak üzere birçok genom projesi yapılmış ve yapılmaya devam etmektedir. İnsan Genom Projesine 1990 yılında başlanmış ve 2003 yılında

sonlandırılmıştır. Bu proje ile insan genomuna ait 3,3 milyar nükleotit baz dizisi ve tüm genlerin belirlenmesi amaçlanmıştır. Geliştirilen projeler ile birlikte eldeki veriler gün geçtikçe artmış ve genomik veritabanı adı verilen bilimsel veritabanları ortaya çıkmıştır. Bunlardan en çok bilinenleri EBI(EMBL-BANK), NCBI(GenBank), CIB(DDJB)'dir. Bu üç farklı veritabanı merkezinin birleşimi ile "Uluslararası Nükleotit Dizi Veri Tabanı Birliği" kurulmuştur.

İGP'nin tamamlanmasından sonra yeni projeler ortaya çıkmıştır. Bunlardan HapMap insan genomunun ancak % 0.1 oranında farklılık gösterdiğini belirtmiştir. Bu proje ile DNA dizi varyasyonlarının modelinin ve haplotip haritalarının ortaya konması amaçlanmıştır. Teknolojinin gelişmesiyle insan genom sayısı İGP'den sonrada artmaya devam etmiştir. Genom varyasyon boyutunun netleştirilebilmesi için büyük boyutlu genom dizilim verisine ihtiyaç duyulmuş ve 1000 genom projesine başlanmıştır. Projenin amacı farklı popülasyonlara ait genomlar arasındaki farklılıkların ortaya çıkarılması olmuştur. Ayrıca Hollanda, Çin, Kore, İrlanda gibi ülkeler kendi ulusal gen projelerini başlatmıştır. Türkiye'de ise genomik çeşitliliğin keşfi için Türkiye Genom Projesi başlatılmış ve Avrupa, Asya, Afrika ile Türkiye arasındaki genetik ilişki ve Türkiye'nin varyasyon haritası ortaya çıkarılmıştır (Alkan C. ve ark., 2017).

Genomik veriler temelde DNA, RNA ve protein dizilerinden oluşmaktadır. DNA ve RNA nükleotit, protein ise amino asit denilen bileşiklerden oluşur. DNA dizileri A, T, G ve C harfleriyle ifade edilen nükleotitlerin bir araya gelmesinden oluşurken, RNA dizilerinde DNA'dan farklı olarak T yerine U nükleotidi vardır. Protein dizileri ise 20 amino asit bileşiğinin farklı sıralarda bir araya gelmesinden oluşur. Ayrıca genlerin ifade düzeylerinin ölçülmesinde kullanılan mikro dizi olarak ifade edilen sayısal tipte verilerde genomik veriler arasındadır. DNA ve proteinin gösterimi Şekil 1.1'deki gibidir.



Şekil 1.1 (a) DNA görünümü (b) Protein görünümü

Bilgisayar bilimlerinin biyoenformatik alanındaki görevi protein dizinlerinin fonksiyonlarını ve yapılarını herhangi bir fiziksel deney gerçekleştirmeden geliştirilen algoritma ve yazılım ile tespit etmektir. Bir proteinin karakteristik özelliklerini anlayabilmek için dizin verilerine bakarak, bazı veritabanlarında bulunan ve bu dizine büyük ölçüde benzerlik gösteren diğer dizinlerin yapısal ve fonksiyonel özelliklerinin incelenmesi gerekir. Dolayısıyla bu alanda birçok dizin hizalama algoritmaları geliştirilmiştir. Bu algoritmalar bütünsel ve yerel hizalama olarak iki kısımda incelenir. Bütünsel hizalama dizilerdeki tüm karakterlerin hizalanmasını esas alırken lokal hizalama birbirinden oldukça farklı sayıda karaktere sahip dizileri hizalamak için kullanılan bir yöntemdir. En yaygın olarak kullanan algoritmalar Needleman-Wunsch ve Smith Waterman dinamik algoritmalarıdır. Sezgisel hizalama algoritmalarına örnek olarak BLAST ve FASTA verilebilir. Sonuç olarak hangi algoritmanın amacımıza uygun olacağına eldeki verilere bakılarak ve çalışmanın amacına göre karar verilir. Örneğin yüksek doğrulukta eşleme elde etmek gerekiyorsa dinamik algoritmalar tercih edilirken eldeki veri büyük olduğunda bu algoritmaları kullanmak mümkün olmamaktadır ve bu durumda sezgisel algoritmalar tercih edilmektedir. Fakat genomik veritabanların giderek çok büyük boyutlara ulaşmasından dolayı sezgisel algoritmaların da tek başına yeterli olmadığı görülmektedir. Bu yüzden son yıllarda yapılan çalışmalar genomik veritabanların indekslenmesi üzerinedir. İndeksleme, veritabanındaki verinin çoğunu eleyerek sorgu maliyetini düşüren bir yöntemdir. Veritabanlarının indekslenmesi hızlı sonuçlar alınmasına büyük katkı sağlamasına rağmen veritabanı boyutunu çok arttırmaktadır.

Bu tez çalışmasında genomik veritabanları incelenmiş ve bu veritabanlarında etkili arama yapabilmek için günümüze kadar geliştirilen algoritmalar araştırılmıştır. Veritabanlarının giderek büyümesi algoritmaların gün geçtikçe verimliliğini düşürmektedir. Bu sebeple geliştirilen veritabanı indeksleme yöntemleri de araştırılmıştır. Kümeleme, genomik verilerin analizinde çok önemli bir araçtır. Bu çalışmada genomik verilerin kümelenmesine yönelik de yöntemler önerilmiş ve önerilen yöntemlerin sonuçları farklı çalışmalardan elde edilen sonuçlar ile karşılaştırılmıştır. Karşılaştırma sonuçlarına göre önerilen yöntemlerin literatürdeki diğer algoritmalardan daha iyi kümeleme yaptığı görülmüştür. Son olarak yapılan çalışmaları birleştiren gen arama ve karşılaştırma üzerine bilgisayar yazılımı geliştirilmiştir.

Tez çalışması 6 bölümden oluşmaktadır. Tez çalışmasının 2. bölümünde temel biyoenformatik kavramlardan ve genomik veritabanlarından bahsedilmiştir.

3. bölümde gen dizilimleri üzerinde geliştirilen arama, hizalama ve indeksleme algoritmaları incelenmiştir.

4. bölümde genomik verilerin kümelenmesi konusu incelenmiştir.

5. bölümde geliştirilen yazılım ve kullanım prensipleri anlatılmıştır.

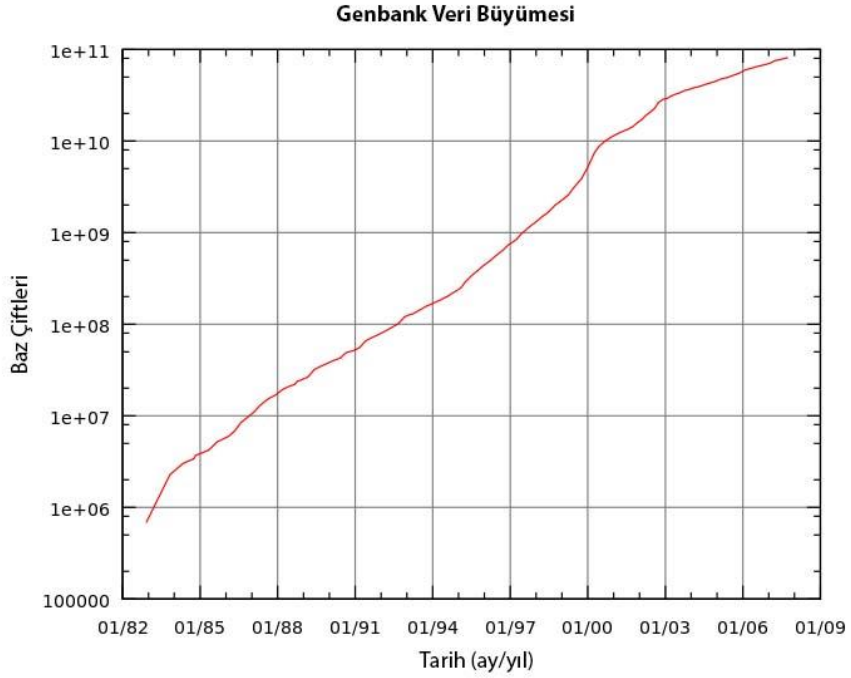
Sonuç kısmında, yapılan çalışmalar özetlenmiştir.

2. TEMEL BİYOENFORMATİK KAVRAMLAR VE GENOMİK VERİTABANLARI

2.1. Biyoenformatik Nedir?

Biyoenformatik, moleküler biyolojinin kavramsallaştırılmasından ortaya çıkan uygulamalı matematik, bilgisayar bilimleri ve istatistiğin bilişim tekniklerinden yararlanarak büyük boyuttaki biyolojik verilerin düzenlenmesi ve anlaşılmasını sağlayan bilim dalıdır. Kısaca biyoenformatik moleküler biyoloji için bilgi yönetim sistemidir ve bir çok pratik uygulama alanına sahiptir (Luscombe ve ark., 2001).

Biyolojik veriler gün geçtikçe inanılmaz oranda artmaktadır (Reichhardt, 1999). Örneğin Ağustos 2000 verisine göre GenBank 8.214.000 nükleotit dizi içermektedir (Benson ve ark., 2000) ve SWISS-PROT protein veritabanı ise 88.166 protein dizisi içermektedir. Haziran 2017 verisine göre ise GenBank 201.663.568 nükleotit dizisi içermektedir ve SWISS-PROT 555,100 protein dizisi içermektedir. GenBank'ın yayınladığı verilere göre 1982'den günümüze kadar baz çiftlerinin sayısı her 18 ayda bir yaklaşık olarak ikiye katlanmaktadır. Çizelge 2.1'de baz çiftleri ve nükleotit dizilerinin sayısı listelenmiştir. Ayrıca Şekil 2.1'de GenBank verilerinin artış grafiği gösterilmiştir.



Şekil 2.1. GenBank verileri artış grafiği

Çizelge 2.1 GENBANK İstatistikleri

	GenBank	
Tarih	Baz Çiftleri	Nükleotit Dizileri
Aralık 1982	680.338	606
Kasım 1983	2.274.029	2.427
Mayıs 1984	3.002.088	3.665
Mayıs 1986	6.765.476	7.416
Şubat 1987	10.961.380	10.913
Haziran 1988	20.795.279	18.226
Mart 1990	40.127.752	33.377
Mart 1992	83.894.652	65.100
Aralık 1993	163.802.597	150.744
Haziran 1995	318.624.568	425.211
Ağustos 1996	602.072.354	920.588
Aralık 1997	1.258.290.513	1.891.953
Nisan 1999	2.569.578.208	3.525.418
Şubat 2000	5.805.414.935	5.691.170
Ağustos 2000	9.545.724.824	8.214.339
Haziran 2002	20.648.748.345	17.471.130
Haziran 2004	40.325.321.348	35.532.003
Ekim 2007	81.563.399.765	77.632.813
Haziran 2014	161.822.845.643	173.353.076
Ekim 2015	202.237.081.559	188.372.017
Aralık 2016	224.973.060.433	198.565.475
Şubat 2017	228.719.437.638	199.341.377
Nisan 2017	231.824.951.552	200.877.884
Haziran 2017	234.997.362.623	201.663.568

Biyoenformatiğin ana konuları aşağıdaki başlıklarda toplanabilir (Telefoncu ve ark., 2003):

- Genom Projeleri : Genome sekansı ve gen haritalarının çıkarılması
- Fonksiyonel Genomik : Mikrodizi gen ekspresyon analizi ve Farmakogenomik
- Yapısal Genomik : Fonksiyonel konfarmasyonun anlaşılması
- Proteomik : Proteinlerin tüm bileşenlerinin analizi
- Kıyaslamalı Genomik : Gen keşfi, fonksiyon tahmini ve tür oluşumunun mekanizması

- Matematiksel Biyoloji : Matematikle ilişkili biyolojik süreçler ve sistemler için istatistiksel modelleme
- Mikrodiziler

Biyoenformatiğin önemli uğraşı alanları aşağıdaki gibidir (Luscombe ve ark., 2001):

1. Biyolojik verileri araştırmacıların erişebileceği ortamlarda saklayabilecek, verileri düzenleyebilecek ve yeni gelen verileri var olanların yanına ekleyebilecek algoritma ve yazılımlar geliştirmek
2. Nükleotit ve amino asit dizilerini, protein bölgeleri ve yapılarını kapsayan farklı tipteki verileri analiz etmek ve yorumlamak
3. Farklı tiplerdeki biyolojik bilgilerin etkin kullanımı ve yönetilmesi için yeni araçlar geliştirmek ve bu araçları kullanmak

Bu tez çalışmasının amacı genomik veritabanlarının araştırılması, veritabanlarındaki büyük boyuttaki verilere erişim, veri tiplerinin anlaşılması, etkili arama yöntemlerinin incelenmesi, yeni yöntemlerin ve ayrıca genomik verilerin analizi için bilgisayar yazılımlarının geliştirilmesidir. Bu yüzden yapılacak bu çalışma biyoenformatiğin kapsamına girmektedir ve bu alana katkı yapacağı düşünülmektedir.

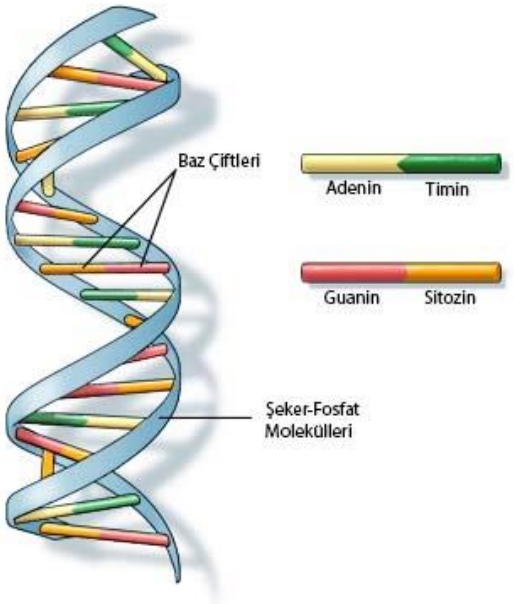
2.2. Genetik Kavramlar

Bir organizmanın onu oluşturan genleri ile birlikte tüm DNA'sının toplamına genom adı verilir. Genom bir organizmanın yaşamı boyunca tüm yaşamsal aktivitelerini ve yapısını belirler. Bir canlının ana genetik materyali deoksiribonükleik asit (DNA) dir.

2.3. DNA

DNA'nın yapısı ilk kez James Watson ve Francis Crick tarafından 1953 yılında keşfedilmiştir. Yapısı Şekil 2.2'de görüldüğü gibi ince, uzun ve çift sarmal yapıdadır. Kimi virüsler dışındaki tüm canlılarda kalıtsal bilgiler DNA'lar aracılığı ile taşınır. DNA ökaryot canlıların hücre çekirdeğindeki kromozom adı verilen yapıların içinde bulunur (Lesk, 2002). Canlının türüne göre hücrede bulunan kromozom sayısı farklılık gösterir. Örneğin insan hücresinde 23 çift kromozom mevcuttur ve 23. kromozom cinsiyeti belirleyen kromozomdur. X kromozomu bayan cinsiyetini ifade ederken Y kromozomu erkek cinsiyetini ifade eder.

DNA, nükleotit denilen küçük moleküllerin birleşiminden oluşur. Her nükleotit bir beş-karbonlu şeker, bir fosfat grubu ve bir organik bazdan oluşur. Sahip olduğu baza göre nükleotitler 4 çeşittir. Bunlar Adenin (A), Sitozin (C), Guanin (G) ve Timin (T)'dir. Yani DNA'yı 4 çeşit harfin farklı sayı ve sıralarda birleşiminden oluşan bir karakter dizisi olarak düşünebiliriz. Bu dizilim sırasına DNA dizilimi de denir. İnsan DNA'sı yaklaşık olarak 3 milyon baza sahiptir. A nükleotidi sadece T nükleotidi ile bağlanırken C nükleotidi sadece G nükleotidi ile bağlanır.



Şekil 2.2 Çift sarmal yapılı DNA

DNA'nın temel olarak iki görevi vardır:

1. Kalıtımı sağlamak için hücre bölünmesi esnasında kendini eşleyerek DNA'nın replikasyonu denilen olayı gerçekleştirmek,
2. Hücrenin bütün kimyasal reaksiyon ve aktivitelerini yöneterek yapı ve fonksiyonunu belirlemektir.

2.4. RNA

RNA yani ribonükleik asit genetik bilginin taşınması için kullanılan önemli bir moleküldür. DNA ile aynı yapıdadır fakat DNA'dan farklı olarak tek iplik ve sarmal yapıda değildir. Ayrıca DNA'da ki timin (T) nükleotidi yerine RNA'da urasil (U) bulunur. RNA'nın farklı türleri vardır. Bunlar hücrede faklı işlevlere sahip olan mesajcı RNA (mRNA), taşıyıcı RNA (tRNA) ve ribozomal RNA

(rRNA)'dır. RNA'nın ortak görevi DNA'dan aldığı bilgiyi taşımak ve protein sentezinin gerçekleşmesini sağlamaktır.

2.5. Gen

Yaşayan bir organizmanın bir bütün olarak karakterini belirleyen en temel kalıtsal birimi gendir. Gen, fiziksel olarak DNA dizi parçasıdır ve RNA aracılığı ile protein adı verilen hücrenin tüm aktivitelerinin gerçekleşmesinden sorumlu moleküllerin yapısını belirler.

İnsanda yer alan 23 kromozom toplamda 20.000-25.000 arası gen barındırır. Bir genin başlangıç bitiş noktası belirli diziler tarafından belirlenir. Genin proteini kodlayan parçalarına ekzom ve ekzomlar arasında kalan hiçbir fonksiyona sahip olmayan bölgelere intron denir. Bir gen diğer genler ile birleşip farklı işlevlere sahip olabilir veya yapısında mutasyon adı verilen bazı değişlikler meydana gelebilir. Bir organizmanın sahip olduğu tüm genlerin kümesi genotip olarak adlandırılır. Her bir gen bir organizmada fizyoloik ve morfolojik işleve sahiptir. Genotipin çevresel etkiler ile birlikte canlının dış görünüşüne yansımasına fenotip denir. Bir genin aynı özelliği kodlayan ve fenotipte farklılık yaratmasına sebep olan varyasyonlarına alel genler denir. İnsanlar biri anneden biri babadan olmak üzere iki kromozoma sahiptir. Bu yüzden insanlar her bir gen için iki alel gene sahiptir. Genler toplam DNA'nın yaklaşık olarak % 1.5 oranına karşılık gelir ve DNA'nın kalan kısmının bilinen herhangi bir işlevi yoktur. Organizmadaki genom büyüklüğü ile o organizmanın gelişmişliği arasında bir ilişki kuramayız. Çünkü bazı tek hücreli organizmalar insanlardan daha geniş genom büyüklüğüne sahip olabilmektedir.

2.6. Protein

Proteinler hücrenin büyük molekülleridir ve genler tarafından kodlanmış görevleri yerine getirir. Bu görevlerden bazıları vücuda giren virüs ve bakteri gibi yapancı partiküllere karşı antikor üretmek, enzim olarak kimyasal reaksiyonlarda katalizör görevi görmek ve farklı hücre doku ve organ arasındaki biyolojik süreçlerde haberleşmeyi sağlamaktır.

Proteinler peptid bağı ile bağlanmış amino asit dizilerinden oluşur. 20 farklı amino asit vardır. Bu amino asitler Çizelge 2.2'de listelenmiştir. 3 nükleotit bir araya geldiğinde kodonu oluşturur. Her bir kodon da bir amino aside karşılık gelir.

Proteinler amino asit dizi kodlarının bir araya gelmesiyle oluşan karakter dizileri ile ifade edilir. Özetle DNA ve proteinler arasında çok sıkı bir ilişki vardır. Bir protein üretmek için hücre DNA dizisinden kodon isimli üç karakter okur ve o kodona karşık gelen amino asit bulunur. Örneğin AAG kodonu bulunduğunda lysine amino asiti üretilir. Amino asitler ve kodonlar arasındaki bu ilişki genetik kod olarak bilinir. Genetik kodda genin sonlandığını gösteren üç farklı kodon vardır. Bir gen DNA ipliği boyunca bir protein şifreler. Her DNA parçacığı protein sentezlemez. Bunlar "önemsiz DNA" olarak bilinir. İnsan DNA'sının % 90'dan fazlası işlevsizdir.

Gen üzerinde yer alan genetik veriden hareketle, proteinin oluşturulması sürecine "gen ekspresyonu" adı verilmektedir (Setubal ve Meidanis, 1997). Tezin ilerleyen kısımlarında gen ekspresyonlarından daha ayrıntılı bahsedilecektir.

Çizelge 2.2 Amino asitlerin isim-kod listesi

Kod	Kısaltma	İsim	Kod	Kısaltma	İsim
A	Ala	Alanine	M	Met	Methionine
C	Cys	Cysteine	N	Asn	Asparagine
D	Asp	Aspartic Acid	P	Pro	Proline
E	Glu	Glutamic Acid	Q	Gln	Glutamine
F	Phe	Phenylalanine	R	Arg	Arginine
G	Gly	Glycne	S	Ser	Serine
H	His	Histidine	T	Thr	Threonine
I	Ile	Isoleucine	U	Val	Valine
K	Lys	Lysine	K	Trp	Tryptophan
L	Leu	Leucine	Y	Tyr	Tyrosine

2.7. Mutasyon

Mutasyonlar genetik koddaki çeşitli nedenlerden dolayı meydana gelen değişimlerdir. Örneğin durma kodonu olarak UAA dizisinde nokta mutasyonu ile A nükleotidi C nükleotidine dönüşürse Serine amino asitini sentezleyen UCA dizisine dönüşür. Bu durumda oluşacak protein hiçbir fonksiyona sahip olmayabilir ya da değişen amino asit durumuna bağlı olarak istenmeyen sonuçlar doğurabilir.

Eğer değişim nükleotit değişimi ise bu mutasyona nokta mutasyonu denir. Eğer dizide eksilme meydana geldiyse bu mutasyona eksilme (deletion) mutasyonu dizide araya girme meydana geldiyse bu mutasyona artma mutasyonu (insertion) denir. Güneş ışığı, radyasyon, X-ray dalgaları, ultraviyole ışınları ve virüslerin DNA'nın yapısını bozması gibi sebeplerden mutasyonlar oluşabilir.

2.8. Başlıca Genomik Veritabanları

Veritabanı yapılandırılmış verilerin topluluğudur ve biyoenformatik alanında milyonlarca veritabanı mevcuttur. Bu veritabanların çoğu diğer veritabanlarından alınan ham verilerin filtrelenip yeni formata dönüştürülmesi ile oluşturulmuştur. Bazı veritabanları özel şirketlere aittir ve erişim ücretlendirilmiştir. Buna rağmen birçok veritabanı herkes tarafından erişime açıktır. Genomik veritabanlarını aşağıdaki başlıklar altında sınıflandırabiliriz:

1. Nükleotit Veritabanları
2. Protein Dizi Veritabanları
3. Mikrodizi Veritabanları

Biyolojik veritabanlarının oluşturulmasında ilk gereksinim İnsan Genom Projesi ile olmuştur. İnsanın kromozomlarının DNA dizilerinin belirlenmesi amacıyla yapılan bu proje yeni teknolojilerin geliştirilmesi ve hesaplamalı biyolojinin doğuşunu sağlamıştır. Günümüzde artık sadece yeni veriler oluşturulmakla kalmayıp aynı zamanda verilere anlam kazandırılıp bilgi haline dönüştürülmeye çalışılmaktadır. Elde edilen bilgiler ile proteinlerin işlevlerinin, yapısının ve evrimsel gelişiminin ortaya çıkarılması sağlanmaktadır.

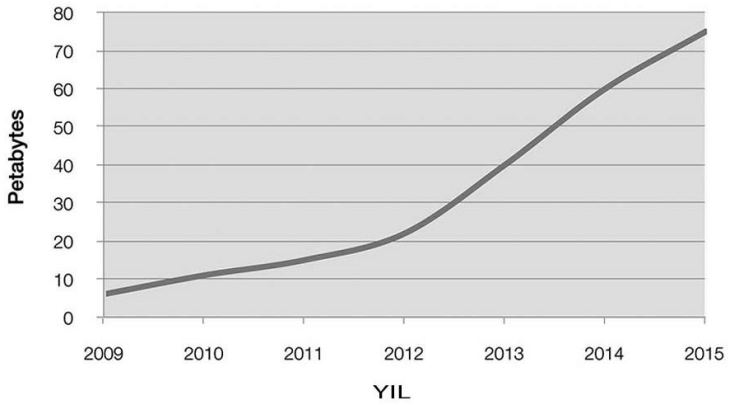
DNA dizilimlerinin hangi parçalarının geni ifade ettiği ve proteinlerin birbiriyle ilişkilerinin anlaşılması gibi pek çok konuya açıklık getirilmeye çalışılmaktadır.

2.9. Nükleotit Veritabanları

Nükleotitler için birçok farklı veritabanları vardır. Bunlardan en önemlileri "National Center for Biotechnology Information" (NCBI) (http://www.ncbi.nlm.nih.gov/genbank), "European Molecular Biology Laboratory-European Bioinformatics Institute" (EMBL-EBI) (Kneale ve Kennard, 1984) ve DNA Databank of Japan (DDJB) (Tateno ve ark., 2002) veritabanlarıdır. NCBI ayrıca GenBank olarak da adlandırılmaktadır. NCBI veritabanı primat,

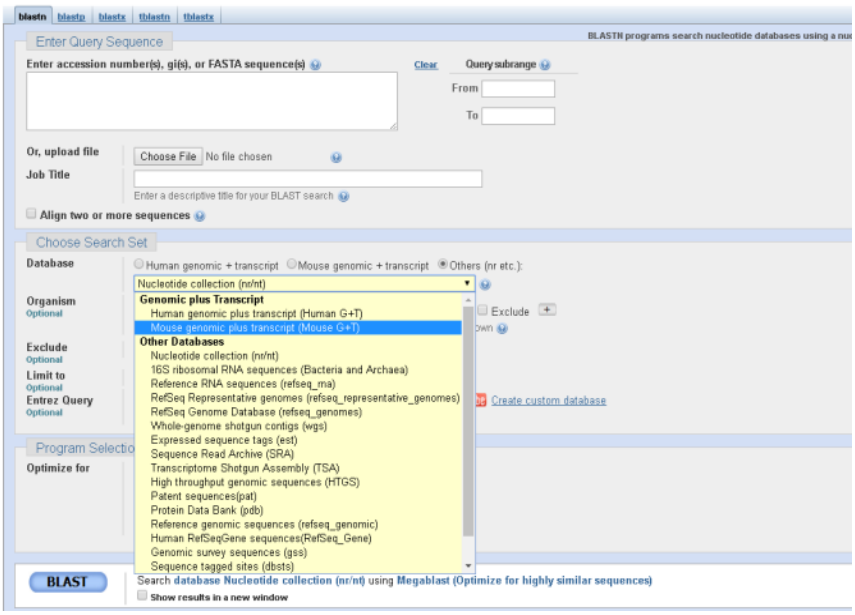
bitkiler, memeliler ve bakteriler gibi çeşitli organizmaların DNA dizilerini içermektedir. Diziler çeşitli araştırma projeleri ve laboratuvarlardan elde edilmiştir. Veri hacmi daha önceki bölümde Şekil 2.1 ve Çizelge 2.1'de gösterildiği gibi çok büyük boyuttadır.

EMBL-EBI veritabanı ise dünyanın ilk nükleotit veritabanıdır ve araştırmacıların katkılarıyla desteklenmeye devam etmektedir. EMBL-EBI veritabanının yıllara göre boyutunu gösteren grafik Şekil 2.3'de gösterilmiştir.



Şekil 2.3. EMBL-EBI Veritabanı Boyutu Artış Grafiği

Veritabanları tüm kullanıcıların erişimine açıktır ve BLAST (Basic Local Alignment Search Tool) (Altschul, 1990) gibi araştırmacıların hizmetine sunulan kullanıcı dostu bioinformatik araçlara sahiptir. Şekil 2.4'de BLAST ile web üzerinden taranabilecek nükleotit veritabanları ve BLAST ekran görüntüsü verilmiştir.



Şekil 2.4. BLAST nükleotit veritabanları web ekran görüntüsü

Bazı nükleotit veritabanlarının açıklamaları aşağıdaki çizelgede verilmiştir.

Çizelge 2.3. Nükleotit Veritabanlarının Açıklamaları

Veritabanı	**Açıklama**
Nr/nt	Bu veritabanı seçildiğinde diziler tüm GenBank, EMBL, DDJB ve PDB veritabanlarında taranır.
refseq_mrna	Bu veritabanı seçildiğinde diziler "Reference Sequence Project" adlı projedeki mRNA dizilerinde taranır.
refseq_genomic	Bu veritabanı seçildiğinde diziler "Reference Sequence Project" adlı projedeki genomik sekanslarda taranır.
Est	Bu veritabanı seçildiğinde diziler transkribe olmuş cDNA'nın kısa alt dizinlerinde taranır.
est_human	Bu veritabanı seçildiğinde diziler EST'nin insan genomu alt kümesindeki dizilerde taranır.
est_mouse	Bu veritabanı seçildiğinde diziler EST'nin fare genomu alt kümesindeki dizilerde taranır.
est_others	Bu veritabanı seçildiğinde diziler EST'nin fare ve insan dışındaki genomların alt kümesindeki dizilerde taranır.
Gss	Bu veritabanı seçildiğinde diziler Genom Tarama Dizisi içinde taranır.
Htgs	Bu veritabanı seçildiğinde diziler henüz tamamlanmamış genomik diziler içinde taranır.
Pat	Bu veritabanı seçildiğinde diziler GenBank'ın patent bölümündeki nükleotit dizileri içinde taranır.
alu_repeats	REPBASE'de Alu tekrarlarını seçer ve sorgulanan dizideki Alu tekrarlarını maskelemek için uygundur.
chromosome	Bu veritabanı seçildiğinde diziler NCBI'dan gelen tüm bir genom ve tüm kromozomlarla kıyaslanır.
Wgs	Bu veritabanı seçildiğinde diziler tüm genomun Shotgun dizilerinin toplamı olan dizilerde taranır.
env_nt	Bu veritabanı seçildiğinde diziler çevresel örneklerden (topraktan veya denizden) gelen diziler içerisinde taranır.

Araştırmacılar web tarayıcıları ile bu araçları kullanabilmekte, API'ler sayesinde kendi araçlarını geliştirebilmekte ve diğer veritabanları ile çapraz bağlantılar kuran ölçeklenebilir arama teknolojilerinden yararlanabilmektedir.

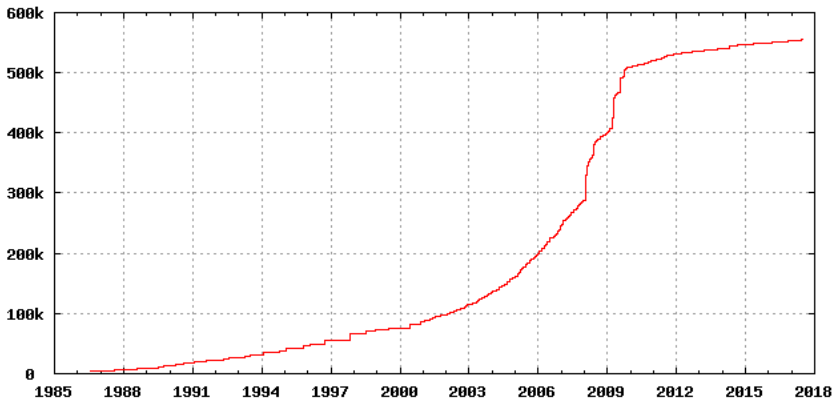
2.10. Protein Veritabanları

Protein veritabanları günümüzde modern biyolojinin en önemli parçası haline gelmiş durumda. Büyük boyutlarda protein yapılarını, işlevlerini ve dizilerini içeren veriler üretilmektedir. Bu veriler bilgisayar tabanlı veritabanları olmadan yönetilemez durumdadır. Yeni bir protein çalışmasında ilk aşama var olan veritabanlarında protein hakkında arama yapmaktır. Yeni protein diğerleri ile karşılaştırılarak proteinin diğer proteinler ile olan ilişkisi bulunur ve böylece protein hakkında daha fazla bilgiye sahip olunur. Araştırmacıların çoklu veritabanları ile çalışması proteinin evrimini, yapısını ve işlevini anlamasına yardımcı olur.

Protein veritabanları internet üzerinden kullanıcı dostu arayüzleri ile erişilebilir durumdadır. Bu veritabanların arasında PIR (Protein Identification Resource) (McGarvey ve ark., 2000) ve UniProt (Bairoch ve Apweiler, 1999) en yaygın olarak kullanılan veritabanlarıdır. UniProt veritabanı araştırmacılara protein dizilimleri ve işlevleri hakkında bilgi sunar. UniProt altında proteinler hakkında açıklamalar içeren UniProtKB alt veritabanı mevcuttur (Xu, 2012).

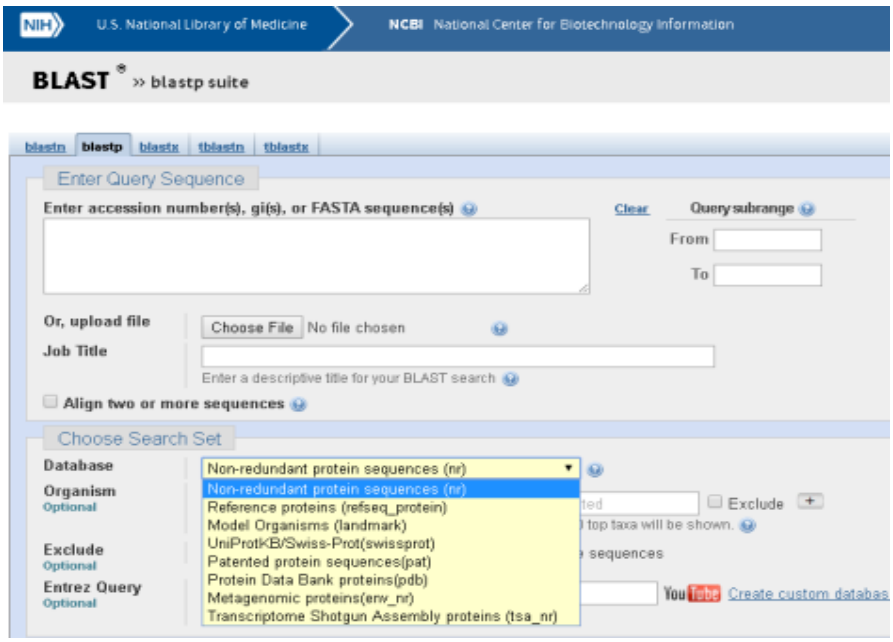
UniProtKB üç bileşene sahiptir. Bunlar proteinler hakkında araştırmacıların elde ettiği verilerin manual eklenmesi ile oluşan SwissProt (Bairoch ve Apweiler, 2000) araştırmacıların hızlı benzerlik araştırmaları yapabilecekleri dizi kümeleri içeren UniRef (Suzek ve ark., 2007) ve tüm protein dizilimlerini ek açıklama içermeden barındıran UniParc (Leinonen ve ark., 2004) bileşenleridir.

Swiss-Prot protein bilgi tabanı olarak 1986 yılında kurulmuştur. Yüksek düzeyde protein fonksiyonunun tanımı, bölgesel yapı, translasyon sonu modifikasyonlar gibi ek bilgiler içerir ve gereksiz dizileri en düşük düzeyde diğer veritabanları ile bütünleşmeyi ise en yüksek düzeyde sunar. SwissProt veritabanının yıllara göre verilerinin artış grafiği Şekil 2.5'de verilmiştir (http://web.expasy.org/docs/relnotes/relstat.html).



Şekil 2.5. UniProtKB/Swiss-Prot yıllara göre girdi sayıları

Veritabanlarının çoğu dizi arama araçları, diğer veritabanlarına çapraz referans bağlantılar, proteinlerin ad ve anahtar kelimelere göre aranması gibi birçok işlevler içermektedir. Bu araçların amacı yeni bir protein üzerinde çalışan araştırmacılara proteinin daha önce keşfedilip edilmediği veya aranan proteine benzer proteinlerin bulmasını sağlamaktır. Böylece araştırmacı aradığı protein dizisinin adı, işlevi, yapısı veya hangi protein ailesine ait olduğu gibi bilgilere erişebilmektedir. Örnek olarak BLAST arama motoru ve tarama yaptığı protein veritabanları Şekil 2.6'da gösterilmiştir.



Şekil 2.6. Protein veritabanı sorgulamada BLAST aracı

Protein veritabanlarının açıklamaları aşağıdaki çizelgede verilmiştir.

Çizelge 2.4. Protein Veritabanlarının Açıklamaları

Veritabanı	**Açıklama**
Nr	Bu veritabanı seçildiğinde diziler GenBank, CDS (genlerin amino asit olarak kodlanan dizileri)'nin protein olarak dönüştürülmüş
Refseq	Bu veritabanı seçildiğinde diziler "Reference Sequence Project" adlı projedeki amino asit dizilerinde taranır.
swissprot	Bu veritabanı seçildiğinde diziler SwissProt veritabanında taranır.
Pat	Bu veritabanı seçildiğinde diziler GenBank'ın patent bölümündeki diziler içinde taranır.
Pdb	Bu veritabanı seçildiğinde diziler protein Veri Bankasında 3 boyutlu yapısı bulunan proteinlerin amino asit dizilerinde taranır.

2.11. Gen Ekspresyon Veritabanları

Gen ekspresyonu işlemi bir gende saklı bulunan kalıtımsal bilgiden yararlanılarak bir gen ürününün, örneğin bir protein veya RNA'nın üretildiği bir işlemdir (Hunter, 1993). Gen ekspresyonu verisi en basit tanımı ile herhangi bir anda bir genin hücre içi aktiflik durumunu ifade eden sayısal bir veridir. Bu veriler ilaç verisiyle birlikte kullanıldığında kişilerin hangi ilaca nasıl yanıt vereceği önceden ön görülebilmektedir (Chengalvala ve ark., 2007).

Gen ekspresyonu üzerine de veritabanı mevcuttur. GEO (Gene Expression Omnibus) bu alanda en sık kullanılan veritabanıdır. GEO, NCBI tarafından 2000 yılında kurulmuş ve araştırmacıların genel kullanımına açılmıştır. Bu veritabanı yüksek çıktı metodolojileri ile üretilmiş gen ekspresyon verileri ve gen ekspresyon seviyesi ölçen veya genomik kazanç ve kayıpları tespit eden mikrodizi tabanlı deney verileri içeren genel bir veri havuzu olarak hizmet eder (Barrett ve ark., 2006). Bu veriler genel olarak genomik DNA ve protein moleküllerini içerir.

3. ARAMA ALGORİTMALARI

Biyolojik veritabanlarında bilgilere erişebilmek ve analiz edebilmek için iyi arama algoritmalarına ihtiyaç vardır. Arama algoritmaları tam motif eşleşmesi ve yaklaşık motif eşleşmesi olarak ikiye ayrılır. Tam motif eşleşmesi olabilmesi için aranan karakter dizisinin mutlaka tam olarak bulunan diziyle eşleşmesi gerekir. Özellikle DNA ve protein dizilerinin yer aldığı veritabanlarında tam motif eşleşmesi teorik olarak önemli olmasına karşın pratikte kullanışlı değildir. Özellikle yanlış dizilenmiş DNA veya protein dizileri tam motif eşleşmesi kullanıldığından bulunamayacaktır. Bu durumda yaklaşık motif eşleşme algoritmalarına ihtiyaç vardır. Fakat bu algoritmalar diğerine göre daha fazla işlem maliyetine sahiptir. Bu yüzden yaklaşık motif eşleşme algoritmaları genel olarak sezgisel algoritmalardır. Yaklaşık motif eşleşmesi algoritmaları hizalama algoritmaları olarak adlandırılır.

3.1. Tanımlar

Tanım 3.1: Belirli sayıdaki sembollerin oluşturduğu kümeye **alfabe** denir.

Tanım 3.2: Alfabenin elemanlarının sonlu sayıda ardışık dizilmesiyle oluşan yapıya **dizin (sequence)** denir.

Tanım 3.3: Bir dizin içindeki bazı elemanların çıkartılmasından oluşan yeni dizine **altdizin (subsequence)** denir.

Tanım 3.4: Bir dizin içerisindeki arka arkaya gelen elemanların oluşturduğu altdizine **altdizi (substring)** denir.

Tanım 3.5: Bir dizinin ilk elemanını da içeren altdizisine **önek (prefix)** denir.

Tanım 3.6: Bir dizinin son elemanını da içeren alt dizisine **sonek (suffix)** denir.

3.2. İkili ve Çoklu Dizi Hizalama

Dizi hizalama (sequence alignment) biyoenformatikte en önemli problemlerden biridir. Verilen iki DNA veya protein dizisi birbirine benzerse bu diziler aynı işleve, yapıya ve evrimsel ilişkiye sahiptir sonucu çıkarılabilir (MountDM, 2004). Evrim teorisine göre farklı türlere ait genomlar mutasyon sonucunda değişime uğramaktadır. Bu genomların benzerliği hesaplandığında evrimsel uzaklıkları öğrenilmiş olur.

DNA dizileri 4 harfin birleşiminden ve protein dizileri 20 amino asitin birleşiminden oluşur.

Dizi benzerliği birçok araştırmacı tarafından çalışılan bir konudur. İki dizinin bir birine dönüştürülmesi için düzeltme mesafesi notasyonunu Levenshtein aşağıdaki gibi açıklamıştır (Levenshtein, 1966):

$X = x_1 x_2 ... x_n$ ve $Y = y_1 y_2 ... y_m$, dizileri $n > m$ iken Σ alfabesi üzerinde iki karakter dizisi olmak üzere, X ve Y dizileri arasındaki benzerliği hesaplamanın genel bir yolu düzeltme işlemlerinin bir dizisi aracılığı ile X dizisini Y dizisine dönüştürmektir.

X dizisini Y dizisine dönüştürmek için üç farklı düzeltme (edit) işlemi vardır. Bunlar; X dizisine uygun bir karakter sembolü ekleme (insertion), X dizisinden bir karakter sembolü silme (deletion) veya X dizisinden bir sembolü Σ alfabesinden bir sembol ile yer değiştirme (substitution) işlemleridir.

$1 \leq i \leq n$ için düzeltme işlemleri formal bir dille aşağıdaki gibi ifade edilebilir:

- Ekleme: x_i den sonra veya önce $s \in \Sigma$ sembolü eklemek
- Silme: x_i sembolünü silmek
- Yer değiştirme: x_i sembolünü $s \in \Sigma$ ile değiştirmek

Değiştirme işlemi aşağıdaki durumlara sahiptir:

- Eğer $s = x_i$ ise eşleşen (matching) değiştirme,
- Eğer $s \neq x_i$ ise eşleşmeyen (non-matching) değişme olarak adlandırılır.

Dizi hizalama iki veya daha fazla dizinin aynı sırada bulunan karakter serilerinin aranması işlemidir. Hizalama işlemi pratik olarak verilen dizilerin alt alta yazılıp dizilere boşluk karakterleri ekleyerek eşit uzunlukta diziler oluşturup birbirleriyle en yüksek oranda eşleşen alt dizileri eşleyebilmektir. Bu eşleme işlemine aşağıdaki örneği verebiliriz:

A=”ACAAGACAGCGT” ve B=”AGAACAAGGCGT” dizileri için;

```
A= A C A A G A C A G  – C G T
     │   │ │   │   │ │   │ │ │
B= A G A A C A  – A G G C G T
```

Örnekte dokuz eşleşen karakter dikey çizgiler ile belirtilmiştir. A dizisinden “C” ile B dizisinden “G” ve A dizisinden “G” ile B dizisinden “C” olmak üzere İki tanede eşleşmeme durumu vardır. Ekleme durumları “–” karakteri ile temsil edilmiş ve son üç karakter iyi bir hizalama sağlamıştır.

Örnekteki eşleşmeme durumları Levenshtein’in düzeltme işlemlerinden yer değiştirme durumuna karşılık gelmektedir. Birinci dizideki boşuk ekleme durumu ve ikinci dizideki boşluk silme durumuna karşılık gelir. Birinci diziyi ikinci diziye dönüştürmek için gerekli işlemler aşağıdaki gibi sıralanabilir:

1. Dizideki ilk “C” karakterini “G” ile değiştir.
2. Dizideki ilk “G” karakterini “C” ile değiştir.
3. Dizideki ikinci “C” karakterini sil.
4. Dizideki son üç karakterden önce bir “G” karakteri ekle.

Bir skor şeması üzerinde iki dizi arasındaki benzerlik skoru hesaplanmak istenirse eşleşen karakterlere artı puan verilirken eşleşmeyen karakterlere eksi puan verilir. Elde edilen tüm değerlerin toplamından elde edilen değer iki dizinin hizalama skoru olur. Eşleşmelere “+1” verirken eşleşmeme ve boşluklara “-1” verilirse örnekte 9.(1) + 4.(-1)=5 skoru elde edilir.

İki dizinin ne kadar benzer olduğu elde edilebilecek en yüksek benzerlik skoruna göre belirlenir. Benzerlik skorunu hesaplayabilmek için benzerlik fonksiyonundan yararlanılır. Verilen benzerlik fonksiyonuna göre en iyi skoru veren hizalama en iyi hizalamadır. *v* ve *w* gibi iki sembol için benzerlik fonksiyonu *d(v,w) v* ve *w* sembollerini hizalamak için verilecek skordur. Benzerlik fonksiyonunun yapısı aşağıdaki gibidir:

$$d(v,w)=\begin{cases}\alpha, \text{ Eğer } v=w\\ \beta, \text{ Eğer v}\neq \text{w}\\ \gamma, \text{ Eğer } v='-' \text{ veya } w='-'\end{cases}$$

Burada α değeri 0'dan büyük iken β ve γ değerleri 0'a eşit veya 0'dan küçüktür.

Bu bölümde iki dizinin bütünsel olarak hizalanması yani global hizalama ve biyologların ilgilendiği iki dizinin birbirine en çok benzeyen alt bölgelerin bulunması yani lokal hizalama ayrıntılı olarak açıklanacaktır. Ayrıca ikiden fazla dizinin hizalanması yani çoklu dizi hizalama hakkında bilgi verilecektir.

3.3. Global Hizalama

Uzunlukları yaklaşık olarak aynı olan dizileri bir bütün olarak hizalamaya global hizalama denir (Mullan, 2006). Global hizalama algoritmalarına örnek olarak en yaygın olarak bilinen Needleman-Wunsch dinamik algoritması verilebilir.

3.3.1. Needleman-Wunsch Algoritması

Dinamik programlama algoritmaları bir problemi alt problemlere bölerek ve bu alt problemleri çözerek en iyi çözümü bulmayı amaçlar. Needleman-Wunsch algoritması en iyi dizi hizalama çözümünü bulmak için Needleman ve Wunsch tarafından geliştirilen dinamik bir algoritmadır (Needleman ve Wunsch, 1970). Needleman-Wunsch algoritmasındaki ana düşünce bir nokta boyunca her alt yol için en iyi yol izlendiğinde yolun kendisinin en uygun çözüm olmasıdır. Böylece en uygun yol alt yolların birleşimi ile elde edilir. İki dizinin hizalanması için Needleman-Wunsch algoritmasını düşünürsek en uygun yol her iki dizinin başlangıcından bitişine kadar genişletilmesi ile elde edilir. Yani bu işlem global hizalamaya karşılık gelir.

Needleman-Wunsch algoritmasını uygulamak için *m* ve *n* uzunluklu iki diziyi karşılaştırmak amacıyla *(m+1)x(n+1)* boyutlu bir skor matrisi oluşturulur. Matrisin her bir *M(i,j)* hücresi dizilerin ayrı ayrı her biri için veya her ikisi için 1

harf eksiğine karşılık gelen üç hücreye (*M(i-1,j), M(i,j-1)* ve *M(i-1,j-1)*) bağlıdır. *M(i,j)* hücresi aşağıdaki formül ile hesaplanır:

$$M(i,j) = max \{ M(i-1,j-1)+S(A(i),B(j)); \quad M(i,j-1) + ekleme(B[j]); \quad M(i-1,j)+ silme(A[i]) \}$$

Burada, boşluklar belirli bir negatif veya sıfır değeri alır ve *S* fonksiyonu eşleşme durumlarında pozitif değer verirken eşlememe durumlarında negatif değer verir.

Matrisin tüm hücrelerinin değeri verilen formüle göre hesaplandıktan sonra en yüksek skoru verecek şekilde matrisin son hücresinden geri izleme yapılır. Böylece hizalama sonucu elde edilir. Geri izlemede çapraz, yukarı veya aşağıya matrisin hücrelerindeki en yüksek değerler takip edilir. Needleman-Wunsch algoritması *O(mn)* zaman karmaşıklığına ve *O(mn)* bellek karmaşıklığına sahiptir.

Örnek 3.1: Aşağıda verilen iki dizide boşluklar için 0, eşleşmeme durumlarında -1 ve eşleşme durumlarında +1 değeri vererek en iyi hizalamayı bulunuz.

A=ACAAGACAGCGT
B=AGAACAAGGCGT

Çözüm 3.1:

Adım 1: A dizisi m boyutlu B dizisini n boyutlu olarak düşünürsek *(m+1)x(n+1)* boyutlarında M matrisi çizilir.

Adım 2: Needleman-Wunsch algoritmasındaki skor değerlerinin atanması prosedürüne göre sol üst köşeden başlayarak matris hücreleri doldurulur.

	-	A	G	A	A	C	A	A	G	G	C	G	T
-	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
A	-1	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
C	-2	0	0	-1	-2	-1	-2	-3	-4	-5	-6	-7	-8
A	-3	-1	-1	1	0	-1	0	-1	-2	-3	-4	-5	-6
A	-4	-2	-2	0	2	1	0	1	0	-1	-2	-3	-4
G	-5	-3	-1	-1	1	1	0	0	2	1	0	-1	-2
A	-6	-4	-2	0	0	0	2	1	1	1	0	-1	0
C	-7	-5	-3	-1	-1	1	1	1	0	0	2	1	0
A	-8	-6	-4	-2	0	0	2	2	1	0	1	1	2
G	-9	-7	-5	-3	-1	-1	1	1	3	2	1	2	1
C	-10	-8	-6	-4	-2	0	0	0	2	2	3	2	1
G	-11	-9	-7	-5	-3	-1	-1	-1	1	3	2	4	3
T	-12	-10	-8	-6	-4	-2	0	0	0	2	2	3	5

Şekil 3.1. A="ACAAGACAGCGT" ve B="AGAACAAGGCGT" dizilerinin global hizalanmasının standart dinamik programlama matrisi ve oklar ile gösterilmiş optimal hizalama

Adım 3: Doldurulmuş matriste son hücreden başlayarak ilk hücreye kadar oklarla belirtilen yol izlenir. Bu yol her bir *M[i,j]* değerinin solundaki, üstündeki ve çaprazındaki değerler ile karşılaştırılmasıyla ve skor belirleme prosedürünü tersten çalıştırarak bulunur. Örneğin *M[i,j] = M[i,j-1]+ekleme(B[j])* eşitliği varsa bu durumda ekleme vardır ve sonraki adımda *M[i,j-1]* hücresinden prosedür devam eder.

Algoritmada izlenecek yolun bulunması *O(m+n)* zaman karmaşıklığına sahiptir. Bazı durumlarda birden fazla yolun izlenebilmesi mümkün

olabilmektedir. Bu durumda çoklu yüksek skora sahip hizalama elde edilmiş olur. Örnekte iki tane optimum skora sahip hizalama mevcuttur.

(1)

A= A C A A G A C A – G C G T
B= A G A A C A – A G G C G T

(2)

A= A C A A G A C A G – C G T
B= A G A A C A – A G G C G T

Dizi hizalamada çoğu zaman dinamik programlama çözümlerini matrisin her bir *(i,j)* hücresini tepe kabul eden *(n+1)x(m+1)* tepeli yönlü bir graf olarak düşünmek yararlı olabilmektedir. Grafın tepeleri arasındaki ayrıtların ağırlıklarının belirlenmesinde aşağıdaki koşullar uygulanır:

- *((i-1, j-1), (i,j))* ayrıtı *S(A(i),B(j))* değeri ile ağırlıklandırılır.
- *((i-1, j), (i,j))* ayrıtı *silme(A[i])* değeri ile ağırlıklandırılır.
- *((i, j-1), (i,j))* ayrıtı *ekleme(B[j])* değeri ile ağırlıklandırılır.

Elde edilen grafta *(0,0)* dan *(n,m)* tepesine kadar maksimum ağırlık elde edecek yolun bulunması iki dizinin hizalanmasına karşılık gelir ve optimum hizalama problemi böylece grafta maksimum ağırlıklı yol bulma problemine dönüşür.

3.4. Lokal Hizalama

Global hizalamanın tersine dizilerin bir bütün olarak değil de alt dizilerinin kendi aralarında hizalanmasına lokal hizalama denir. Lokal hizalamanın amacı alt diziler içindeki saklı alt dizilerin ortaya çıkarılması ve bu diziler hakkında bilgiye erişilmesidir. Lokal hizalama algoritmalarına örnek olarak en yaygın olarak bilinen Smith-Waterman dinamik algoritması verilebilir.

3.4.1. Smith-Waterman Algoritması

1985 yılında T.F. Smith ve M.S. Waterman yerel hizalama yöntemini kullanarak iki amino asit dizisi arasında optimum hizalamayı sağlayan algoritmalarını sunmuştur (Smith ve Waterman, 1981). Algoritmanın temeli Needleman-Wunsch algoritmasına dayanır. İki algoritma arasındaki temel fark matris oluşturulurken negatif değerlerin oluşmasını engellemek için *0* değerinin dördüncü koşul olarak kullanılmasıdır. Oluşturulan matriste ilk satır ve sütundaki hücreler *0* değerlerinden oluşmaktadır. Needleman-Wunsch algoritmasında olduğu gibi amaç maksimum skora ulaşılacak yolu bulmaktır fakat burada ulaşılacak nokta matrisin ilk hücresi olmak zorunda değildir. Onun yerine matriste en yüksek değerden başlayarak *0* değerine ulaşılmaya çalışılır.

Smith-Waterman algoritmasında *M(i,j)* hücresi aşağıdaki formül ile hesaplanır:

$$M(i,j) = max \{ \quad 0;$$

$$M(i-1,j-1)+S(A(i),B(j));$$

$$M(i,j-1) + ekleme(B[j]);$$

$$M(i-1,j)+ silme(A[i]) \}$$

Smith-Waterman algoritması Needleman-Wunsch algoritması gibi *O(mn)* zaman karmaşıklığına ve *O(mn)* bellek karmaşıklığına sahiptir.

Örnek 3.2: Aşağıda verilen iki dizide boşluklar için -2, eşleşmeme durumları -3 ve eşleşme durumları +2 değeri verilerek en iyi lokal hizalamayı bulunuz.

A= ATGCATCCCATGAC

B= TCTATATCCGT

Çözüm 3.2: Smith-Waterman algoritmasındaki skor değerlerinin atanması prosedürüne göre sol üst köşeden başlayarak matris hücreleri doldurulur.

	-	**A**	**T**	**G**	**C**	**A**	**T**	**C**	**C**	**C**	**A**	**T**	**G**	**A**	**C**
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	2	0	0	0	2	0	0	0	0	2	0	0	0
C	0	0	0	0	2	0	0	4	2	2	0	0	0	0	2
T	0	0	2	0	0	0	0	2	1	0	0	2	0	0	0
A	0	2	0	0	0	2	0	0	0	0	2	0	0	2	0
T	0	0	4	2	0	0	2	0	0	0	0	4	2	0	0
A	0	2	0	0	0	2	0	0	0	0	2	0	0	2	0
T	0	0	4	2	0	0	4	2	0	0	0	4	0	0	0
C	0	0	2	0	4	0	0	6	4	2	0	0	0	0	2
C	0	0	0	0	2	0	0	4	8	6	4	2	0	0	2
G	0	0	0	2	0	0	0	2	6	5	3	1	4	2	0
T	0	0	2	0	0	0	2	0	4	3	2	5	3	1	0

Şekil 3.2. A="ATGCATCCCATGAC" ve B="TCTATATCCGT" dizilerinin lokal hizalanmasının standart dinamik programlama matrisi ve oklar ile gösterilmiş optimal lokal hizalama

Doldurulmuş matriste en yüksek değer olan 8'den başlayarak 0 değerine doğru Şekil 3.2'de belirtilen oklar izlendiğinde aşağıdaki hizalama sonucu elde edilir. Sonuç olarak elde edilen hizalama skoru 8'dir.

```
A= A T C C
   | | | |
B= A T C C
```

Örnekten de anlaşılacağı üzere bir genomun benzer alt dizilerini kısmen uzun bir DNA dizisinde araştırmak istendiğinde Smith-Waterman uygun bir algoritma olacaktır. Çünkü algoritma ikili eşleşmeler arasındaki en iyi eşleşen alt diziyi verecektir. Ayrıca milyonlarca nükleotitle alt dizi eşleşmelerinde bütünsel hizalamanın elde edilmesi imkansızdır (Krane ve Raymer, 2002).

3.5. Çoklu Dizi Hizalama

Çoklu dizi hizalama biyolojik dizilerin karşılaştırmalı yapısal ve işlev analizlerinde anahtar role sahip en yaygın işlemlerden bir tanesidir. Genellikle nükleik asit veya protein dizilerinin işlev, yapı, dizi ilişkilerinin değerlendirilmesinde temel bir biyolojik kavramdır. Çoklu dizi hizalama 3 ve daha fazla protein veya nükleik asit dizilerinin birbirleri ile veya daha önce hizalanmış diğer diziler ile kısmen veya tamamen hizalanması işlemidir.

Çoklu dizi hizalama ile elde edilen veriler biyolojik birçok sahada kullanılmaktadır. Çalışılan gen-protein daha büyük gen-protein grubu ile ilişkili ise grup üyesi tüm grubun işlev, yapı ve değişimi ile ilgili bilgiyi taşıyabilir. Yani tüm yapının uzun sürecek deneysel incelenmesine alternatif olarak kısa bir dizinin çoklu hizalanması ile genel yapı ile ilgili fikir sahibi olunabilir. Çoklu dizi hizalaması ile protein ailelerinde mevcut değişim veya korunuma neden olan bölgelerin tespitinde önemli sonuçlar ortaya çıkarılabilir. Kısaca çoklu dizi hizalama genellikle proteinlerin işlev ve yapısal kararlılıklarından sorumlu amino asit kalıntıları gibi potansiyel olarak önemli bölgelerin bulunmasını sağlar. DNA dizilerinde korunmuş bölgeler düzenleyici rolü üstlenebilir (Notredame, 2007).

Verilen k tane $S_1,S_2,\ldots,S_k$ dizisini çoklu hizalama yaparken dizileri aynı uzunluğa getirmek için boşluk karakterleri eklenir.

Örnek 3.3: MQPILLLV, MLRLL, MKILLL ve MPPVLILV dizilerini çoklu hizalayınız.

Çözüm 3.3:

M Q P I L L L V

M L R - L L - -

M K - I L L L -

M P P V L I L V

Diziler arasındaki mesafeyi hesaplamanın birçok yolu vardır. Bunlardan biri ikililerin toplamı (The sum-of-pairs (SP)) skor şemasıdır. Bu yöntemde her kolon için kolonlardaki ikililerin skorlarının toplamı ile hesaplama yapılır. Daha sonra her kolon için bulunan toplam değerlerin hepsinin toplanmasıyla hizalama skoru elde edilmiş olur. Eşleşme durumlarına "+1" , eşleşmeme durumlarına "-1" ve boşluklara "-2" değeri verilecek olursa örneğin 4. kolonunun skorlaması aşağıdaki gibi olacaktır:

SP(I, – , I, V) =

skor (I, –) + skor (I, I) + skor (I, V) + skor (–, I) + skor (–, V) + skor (I, V)

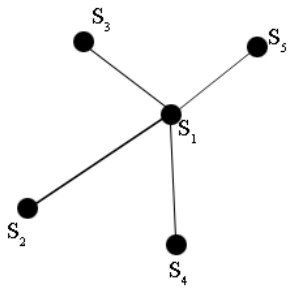
=(-2)+1+(-1)+(-2)+(-2)+(-1)=-7

Çoklu hizalama algoritmalarının büyük bir kısmı çift dizi hizalama işlemine dayanır (Notredame, 2007). Çoklu dizi hizalama algoritmaları dizi özellikleri incelendikten sonra en yakın gözüken iki diziyi hizalar. Elde edilen sonuç ile en yakın üçüncü dizi hizalanır ve sonunda çoklu dizi hizalama elde edilmiş olur. Teorikte Needleman-Wunsch algoritması kullanarak her dizinin diğerleri ile hizalanmasıyla en iyi çözüm elde edilebilir. Fakat bu yöntemi kullanmak çok sayıda dizi hizalanmak istendiğinde efektif olmayacaktır. Örneğin *k* tane dizinin hizalanması için *k(k-1)/2* kere algoritmanın çalışması gerekecektir. Yani algoritmanın zaman karmaşıklığı $O(n^2k^2)$ olacaktır. Bu nedenle çoklu dizi hizalamalarında sezgisel yöntemler tercih edilmektedir. Bu algoritmalardan biri Yıldız Hizalama Sezgiseli (Star Alignment Heuristic) algoritmasıdır.

3.5.1. Yıldız Hizalama Sezgiseli

Algoritmanın temel prensibi tüm dizilere en yakın olan diziyi merkez kabul edip dizileri merkez diziye göre hizalamaktır. Burada merkez dizi diğer diziler

ikili hizalandığında en benzer dizidir. Şekil 3.3'de merkez dizi S_1 dizisidir. Formal olarak merkez diziyi S_c ile gösterirsek merkezi bulmak için $\sum_{i \neq c} sim(S_i, S_c)$ değerinin maksimum olması amaçlanır.



Şekil 3.3. S_1 dizisi merkez dizidir

Örnek 3.4: S_1="ATTGCCATT", S_2 ="ATGGCCATT, S_3="ATCCAATTTT", S_4="ATCTTCTT" ve S_5="ACTGACC" dizilerini yıldız hizalama sezgiseli kullanarak çoklu hizalayınız.

Çözüm 3.4:

Adım 1: İkililerin toplamı skorlama şemasına göre dizilerin birbirleri ile toplam hizalama skorlarının yer aldığı matris oluşturulur.

Çizelge 3.1. Dizilerin İkili Hizalama Skorları

	S_1	**S_2**	**S_3**	**S_4**	**S_5**	
S_1	–	7	-2	0	-3	**2**
S_2	7	–	-2	0	-4	**1**
S_3	-2	-2	–	0	-7	**-11**
S_4	0	0	0	–	-3	**-3**
S_5	-3	-4	-7	-3	–	**-17**
	2	**1**	**-11**	**-3**	**-17**	

Çizelge 3.1'e göre S_1 dizisi kendinden başka diğer dizilere en yakın konumda olduğundan merkez kabul edilir.

Adım 2: S_1 dizisi ile diğer dizilerin ikili hizalamaları sırasıyla aşağıdaki gibi oluşturulur.

S_1= A T T G C C A T T
S_2= A T G G C C A T T

S_1= A T T G C C A T T – –
S_3= A T C – C A A T T T T

S_1= A T T G C C A T T
S_4= A T C T T C – T T

S_1= A T T G C C A T T
S_5= A C T G A C C – –

Adım 3: Son adımda bir önceki adımdan elde edilen hizalamalardan yararlanarak S_1 dizisine göre çoklu hizalama oluşturulur.

S_1= A T T G C C A T T – –

S_2= A T G G C C A T T – –

S_3= A T C – C A A T T T T

S_4= A T C T T C – T T – –

S_5= A C T G A C C – – – –

Algoritmada k tane n uzunluklu dizinin birbirleri ile karşılaştırılması için *k(k-1)/2* hizalama işlemi gerekir. Her bir hizalama O(n^2) zaman karmaşıklığına sahiptir. Öyleyse örnekteki Adım 1'de O(k^2n^2) zaman karmaşıklığına sahiptir. Adım 2'de merkez diziye göre hizalama işlemi O(kn^2) zaman karmaşıklığa sahiptir. Yani algoritmanın toplamda zaman karmaşıklığı O(k^2n^2)'dir.

3.6. İndeksleme

Genomik verilerin saklanmasında doğru veri yapılarını kullanmak verilere erişim hızını arttırmaktadır (Salzberg, 1998). Bunu başarmak için de indeksleme metotları kullanılmalıdır. İndeksleme yapmak basit bir sorgu taramasında tüm veritabanının taranması yerine veritabanının belirli bölümlerinin taramasını sağlar. Bu da arama işlemlerinde hesaplama maliyetini düşürerek sorgu sonuçlarının daha hızlı elde edilmesini sağlar.

Genom dizileri dinamik programlama gibi doğrudan yaklaşımlar için çok büyük boyutlara ulaşmıştır. Bu sebeple indeksleme yöntemleri giderek çok önem kazanmaktadır. Bilgisayar bilimlerinde birçok indeksleme yöntemi vardır. En basit indeksleme yaklaşımı verileri alfabetik veya sayısal olarak sıralamaktır. Bu alanda indeksleme yapmak için inverted (ters) indeks, B-trees ve Sonek Ağaçları gibi belli başlı yöntemler kullanılmaktadır (Manber ve Myers, 1993), (Bayer ve McCreight, 2002), (Manning ve ark**.,** 2008).

DNA dizilerini birer doküman olarak düşünecek olursak aradığımız dizi sorgusunun hangi dokümanda olduğunu anlayabilmek için her birinde arama yapmak gerekir. Oysaki bir indeks sayesinde bize hangi dokümanda hangi terimlerin yani alt dizilerin bulunduğu bilgisi verilse sadece ilgili dökümanlarda arama yapmak yeterli olacaktı. Bize bu bilgiyi verebilecek indeksleme yöntemlerinden biri inverted indeks yöntemidir. Bu yöntemde öncelikle terimler alfabetik olarak sıralanır ve terimlerin karşısına hangi dokümanlarda bulunduğu bilgisi verilir. Ayrıca bu terimlerin dokümanlarda nasıl bir sıklıkla bulunduğu bilgiside saklanabilir. Eğer bazı terimler yüksek sıklıkla her dokümanda bulunuyorsa bu terimler durma kelimeleri olarak adlandırılır. Bu terimler bize çok fazla bilgi vermediğinden bunlar indekslemeden çıkarılır. Ayrıca boşluklar ve noktalama işaretleri de çıkarılır. Yapılan bu işlemlere normalizasyon adı verilir.

Inverted indeks yöntemi için aşağıdaki örneği verebiliriz.

Doküman 0, D0= “ATCG ATT ACC”;

Doküman 1 ,D1= “ATCG ACG AAA ACC”;

Doküman 2, D2 = “ATT ACG AAA ATTC ACC”;

Yukarıda verilen dokümanlar için inverted indeks Çizelge 3.2’de görüldüğü gibi oluşturulmuştur.

Çizelge 3.2 Inverted indeks yapısı

Terim	**Dökümanlar**
ATCG	D0,D1
ATT	D0,D2
ACC	D0,D1,D2
ACG	D1,D2
AAA	D1,D2
ATTC	D2

Bu tabloya göre eğer “ATCG” sorgusu aranırsa cevap olarak D0 ve D1 dokümanları dönecektir.

Eğer “ATCG ACC” gibi bir sorulama yapılırsa her bir sorgu terimi için doküman listelerinin kesişimi alınır.

Index(‘ATCG’) -> {D0,D1}

Index(‘ACC’) -> {D0,D1,D2}

Arama sonucu = {D0,D1} $\cap$ {D0,D1,D2} = {D0,D1}

3.6.1. Sonek Ağaçları

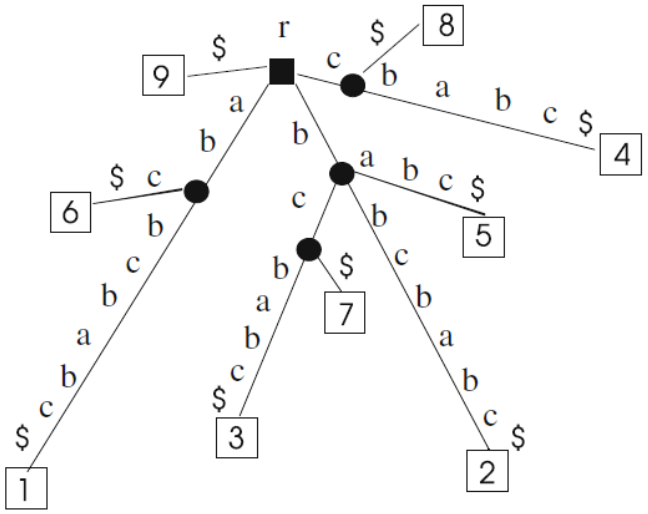
Sonek ağaçları bir dizine ait tüm sonekleri göstermek ve bunlarla ilgili algoritmik problemleri çözmek için kullanılan indeksleme yöntemidir. Sonek ağaçları Weiner tarafından ilk olarak bir karakter dizisinin soneklerini temsil eden bir veri yapısı olarak sunulmuştur (Weiner,1973). Sonek ağacı algoritmalarının bellek karmaşıklığı yüksek olmasının yanında dinamik algoritmalara göre çok daha iyi performans sunar (Gusfield, 1999). Bu indesleme yöntemi tam metin eşleme ve patern eşleme gibi problemlerde biyoenformatik alanında birçok uygulamaya sahiptir. Sonek ağaçları, kökten başlayarak, sembollerden oluşan dizinleri ifade eder ve her düğüm kökten bu düğüme kadar olan yoldaki bütün dizinlerin birleşimi olan dizini temsil eder.

Sonek ağacı bir dizinin tüm soneklerini tutan ve her düğümün en az iki çocuk düğümü olan bir veri yapısı olarak görülebilir. $S = s_1 ... s_n, \Sigma$ alfabesi üzerinde tanımlı *n* sembolden oluşan bir dizin, $ bu alfabede yer almayan özel bir sembol ve s_n bu özel sembole eşit olsun. S dizisine ait T sonek ağacı bu dizinin soneklerini tutan bir yapıdır ve aşağıdaki 4 koşulu sağlar.

1. Ağaç üzerinde kökten her bir yaprağa kadar olan yol, S dizisinin farklı bir sonekini tutar.
2. T'nin her bir kenarı S'nin boş olmayan bir alt dizinini temsil eder.
3. Kök düğüm hariç her bir ara düğümün en az iki çocuk düğümü vardır.
4. Aynı ara düğüme bağlı her kenarın temsil ettiği alt dizin farklı sembollerle başlamalıdır.

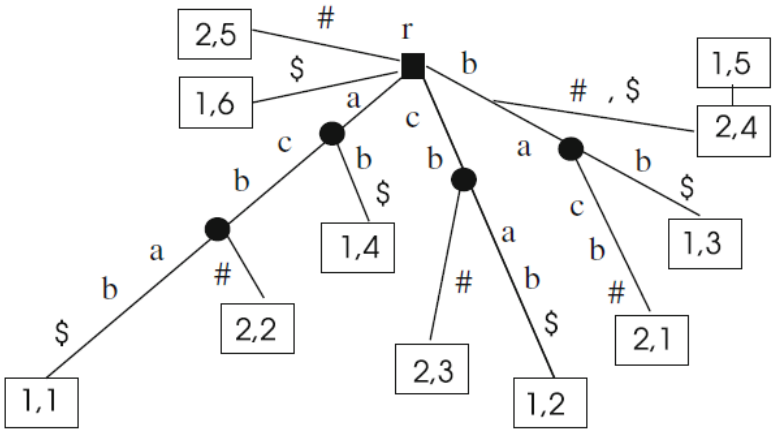
Tanım 3.7 : Bir T sonek ağacı S karakter dizisinin n tane karakterini temsil etmek için 1'den n'ye kadar numaralandırılmış yapraklardan oluşan yönlü bir ağaçtır.

Bir sonek ağaç temsili, her bir S karakter dizisi için doğru bir sonek ağacını garanti etmez. Eğer S karakter dizisinde sonek ağacının öneki onun soneki ile aynı ise ulaşılacak sonek yolu bir yaprak tarafından temsil edilmeyecektir. Bu yüzden S karakter dizisinin sonuna özel bir $ karakteri eklenir ve böylece S karakter dizisinin soneki Şekil 3.4'deki S = "abbcbabc" karakter dizisinin sonek ağacı örneğinde olduğu gibi $ sembolü ile sonlanır.



Şekil 3.4. S = “abbcbabc” karakter dizisi için sonek ağacı örneği. Örnekte yapraklar soneklerin indeksleri ile numaralandırılmıştır.

Genelleştirilmiş bir sonek ağacı iki veya daha fazla karakter dizisi içeren bir yapıdadır. Bu durumda her yaprak Şekil 3.5’de görüldüğü gibi sonekin hangi karakter numarasına ait olduğu ve bu dizide hangi indis numrasından itibaren başladığını belirten iki sayı ile temsil edilir.

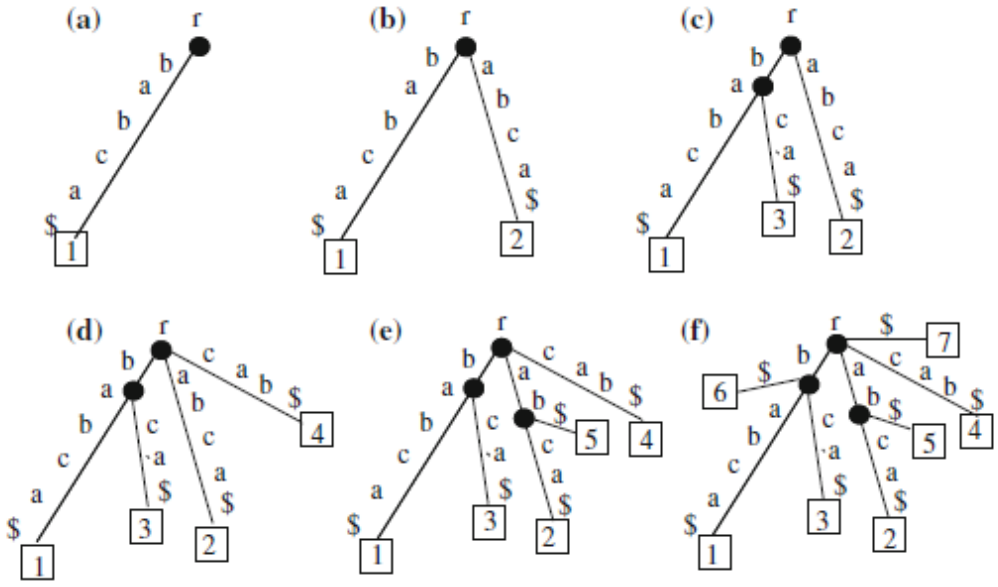


Şekil 3.5. S_1 = “acbab” ve S_2 = “bacb” dizileri için genelleştirilmiş sonek ağacı örneği. Burada yapraklar karakter dizisi numarası ve bu dizide hangi pozisyonda olduğunu belirten indis numarası ile etiketlenmiştir. “$” karakteri S_1 dizisi için durma karakteri iken “#” karakteri S_2 dizisi için durma karakteridir.

3.6.2. Sonek Ağaçları Oluşturma Naive Algoritması

T(S) S karakter dizisinin sonek ağacı olmak üzere S[n] karakter dizisi için başlangıçta S[1, . . . , n$] soneki ağaca eklenir ve T_1 sadece bu sonekten oluşur. Sonrasında 2 ≤ i ≤ n için S[i, n] en uzun sonek ekten en kısa soneke kadar ağaca eklenir. Burada her i+1 adımında T_i ağacı r kökünden başlayarak geçilir ve S[i+1,n] öneki T_i nin bir yoluna eşlenip eşlenmediği aranır. Eğer böyle bir eşleme olmazsa yeni bir (i+1) yaprağı olarak kökten bağlanır ve ağaca eklenir. Eğer eşleme gerçekleşirse eşleşen yolda semboller baştan itibaren eşleşmeyen sembole kadar kontrol edilir. Eğer eşleşmeme (u,v) ayrıtında ise (u,v) ayrıtı yeni bir dala ayrılır aksi durumda eğer eşleşmeme bir w tepesinden sonra ise yeni bir dal w tepesine bağlanır. Sonek ağacı oluşturma algoritması aşağıda verilmiştir.

S = “babca” karakter dizisinin sonek ağacının oluşturulması Şekil 3.6’da gösterilmiştir. Burada öncelikle T_1 oluşturulurken en uzun sonek olan S[1..n]$ r köküne (a)’da olduğu gibi bağlanır. Sonra ağaçta S[2..n]$ ile eşleşen bir önek olmadığından S[2..n]$ yeni bir dal olarak r köküne bağlanır ve böylece (b)’deki T_2 ağacı oluşur. Daha sonra S[3..n]$ sonekinin ilk karakteri ile S[1..n]$ ayrıtındaki ilk karakter eşleştiğinden bu ayrıt (c)’deki gibi ikiye ayrılır. Bu şekilde devam edilerek tüm sonek ağacı oluşturulur.



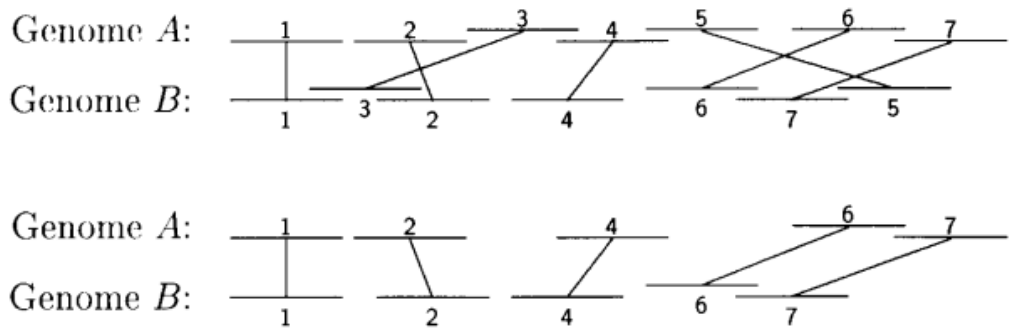
Şekil 3.6. S = “babca” karakter dizisi için sonek ağacının oluşturulması

Sonek ağacının oluşturulması Naive algoritması O(n) bellek karmaşıklığına ve O(n^2) zaman karmaşıklığına sahiptir.

3.6.3. MUMmer Hizalama Algoritması

Genom dizilerinin global hizalaması için geliştirilmiş indekslemeye dayalı bir algoritmadır (Delcher ve ark., 1999) . Temel fikir sonek ağaçları veri yapısı, en büyük artan alt dizin (longest increasing subsequence, LIS) algoritması ve Smith-Waterman hizalama algoritmasınn uygulanmasıdır. Algoritmanın en önemli özelliği büyük boyutlu genomik dizilerin hizalanmasında başarılı olmasıdır.

Algoritmada öncelikle maximal unique match olarak ifade edilen iki dizinde de aynı olan en uzun diziler sonek ağaçları yardımıyla bulunur ve bir listeye eklenir. Daha sonra bulunan unique match'ler birinci dizideki bulundukları sıraya göre Şekil 3.7'de görüldüğü gibi numaralandırılır. LIS algoritmasını kullanarak her iki dizinde de aynı sıralamada olan maximal unique match dizileri birbirleriyle eşlenir. Örneğin ikinci dizindeki sıralama {1, 2, 10, 4, 5, 8, 6, 7, 9, 3} şeklinde ise LIS algoritması sonucu {1, 2, 4, 5, 6, 7, 9} şeklinde olur. Şekil 3.7'de 3 ve 5 numaralı maximal unique match'ler B dizisinde farklı bir sırada olduğundan bu maximal unique match'ler listeden çıkarılır. Çıkarılan bu alt diziler daha sonra Swith Waterman lokal hizalama algoritması ile hizalanacaktır.



Şekil 3.7. A ve B genomlarının maximal unique match'ler bulunduktan sonra numaralandırılması ve LIS algoritması ile eşlenmesi

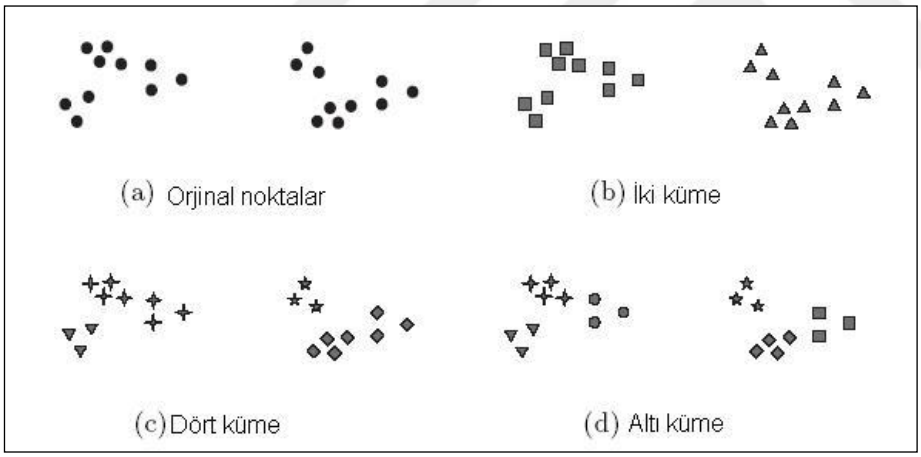
MUMmer hizalamada LIS algoritması *O(K log K)* zaman karmaşıklığına sahiptir. Burada *K*, maximal unique match'lerin sayısıdır. Etkili bir sonek ağaçları oluşturma algoritması *O(n)* zaman karmaşıklığına sahiptir. Dolayısıyla $O(n^2)$ zaman karmaşıklığına sahip dinamik dizi hizalama algoritmalarına göre MUMmer algoritması büyük boyulu genomik diziler için çok daha hızlı sonuçlar üretebilmektedir.

4. GENOMİK VERİLERİN KÜMELENMESİ

Kümeleme temel olarak nesnelerin bazı benzerlik ölçümlerine göre gruplanması işlemidir. Her kümeleme yönteminin amacı aynı kümeye ait olan nesnelerin diğer analiz edilmiş nesnelere göre daha çok benzer olmasıdır. Kümeleme bilgisayar bilimlerinde en çok çalışılan konular arasındadır.

Kümeleme analizi verilerin özelliklerini göz önüne alarak birbirleri ile benzer olan verileri alt kümelere ayırmayı sağlayan çok boyutlu veri analiz yöntemidir. Kümeleme analizi, nesneleri küme içerisinde çok benzer biçimde, kümeler arasında farklı olacak biçimde kümeler. Tıp, biyoenformatik, ekonomi, bankacılık, mühendislik, astronomi ve yerbilim, sosyal bilimler vb. birçok alanda önemli uygulamalara sahiptir.

Nesneleri kümelere ayırmanın birçok yolu olduğundan uygulamalarda kümeleme işlemleri karmaşıklığa sebep olmaktadır. Örneğin Şekil 4.1'de 20 değişik nokta ve bu noktaları kümelere ayırmak için üç farklı yol gösterilmektedir.



Şekil 4.1. Aynı noktalardan oluşan bir setin değişik yollarla kümelenmesi

İki çeşit klasik kümeleme metodu mevcuttur. Bunlar hiyerarşik kümeleme ve hiyerarşik olmayan kümelemedir. Hiyerarşik kümeleme algoritmaları her bir noktayı tekil bir küme olarak ele alır, sonrasında ardışıl olarak en yakın iki kümeyi birleştirir. Bu işlemler tek ve tam küme kalıncaya kadar devam eder. Hiyerarşik olmayan kümelemede ise belirlenen küme sayısı kadar verilerin kümelere ayrılmasını amaçlar. K-means algoritması (Mac Queen, 1967) hiyerarşik olmayan kümeleme algoritmaları arasında en çok bilinen ve en popüler olan kümeleme algoritmasıdır (Theodoridis ve Koutroumbas, 2006). Algoritmanın basitliğinden dolayı çok yaygın uygulama alanına sahiptir.

K-means kümeleme algoritması n tane noktanın önceden belirlenmiş k değerine gore m boyutlu uzayda k tane ayrık kümeye ayrılması işlemidir. Burada aynı kümeye ait noktalar, birbirlerine diğer kümelere ait noktalara göre daha çok benzerdir.

Son yıllarda teknoloji ile birlikte biyolojik verilerde büyük bir artış vardır. Bununla birlikte bu verilerin anlamlı gruplara ayrılması biyoenformatiğin temel araştırma alanlarından biridir. Bu veriler genel olarak DNA dizileri, genom, protein amino asit dizileri, ifade olmuş gen dizileri (cDNA) ve gen ekspresyonu verileridir.

Protein dizilerini gruplamak için dizilerin benzerliğini bulmada dizi hizalama metotlarından yararlanılır. Bu metotlar iki dizi arasındaki minimum sayıda silme, ekleme ve yer değiştirme kullanarak hizalamayı amaçlar. Hesaplanan hizalama skoru diziler arasında benzerlik ölçütü olarak kullanılır. Yine de benzerlik ölçütü için önceki bölümde bahsedildiği gibi kullanılan farklı yaklaşımlar da mevcuttur.

Biyolojik dizilerin kümelenmesindeki amaç diziler arasındaki evrimsel ilişkinin araştırılması ve fonksiyonlarının belirlenmesidir. Böylece hastalıkların anlaşılması ve tedavi yöntemlerinin geliştirilmesinde büyük katkı sağlayacaktır.

Biyolojik dizilerin kümelenmesinde global hizalama kullanmak ilk bakışta doğru tercih olarak gözükmektedir. Fakat proteinler domain olarak adlandırılan amino asit zincirlerinden oluşur ve global olarak çok farklı olan iki protein dizisi benzer domainlere sahip olabilir. Bu durumda bu proteinlerin benzer fonksiyonlara sahip olmasını sağlar. Global hizalama bu proteinleri farklı kümelere yerleştirirken lokal hizalama proteinleri aynı kümeye yerleştirir. Benzerlik değeri lokal veya global olarak hesaplandıktan sonra herhangi bir hiyerarşik veya kısmi kümeleme yöntemi kullanılabilir. Bu yöntemler polinom zamanlı olmasında rağmen büyük biyolojik veri hacmi bu yöntemleri kullanışsız hale getirir. Bu nedenle araştırmacılar bu dizilerin bazı özelliklerini kullanarak kümelemeye çalışır. Bunlardan biri de verilerin indekslenmesidir.

Genomik verilerin kümelenmesinde amaç birbirine benzer DNA/RNA protein dizilerini gruplamak ve böylece aralarındaki filogenetik ilişkileri ve fonksiyonlarını belirlemektir. Ayrıca hastalıkların tanısında ve tedavisinde geliştirilen uygulamalarda kümeleme kullanılmaktadır. Genomik dizilerin

kümelenmesinde benzerlik ölçümü için dizi hizalama algoritmaları ve BLAST gibi araçlardan yararlanılır. Gen ekspresyon verilerinin kümelenmesinde ise benzerlik ölçümü için en sık kullanılan ölçme aracı Öklid uzaklığıdır. Mikrodizi deneyleri sonucunda elde edilen büyük boyutlarda ki verilerin analiz edilerek yorumlanması ve böylece biyolojik olarak anlamlı olan bilgilere ulaşılması gerekmektedir. Gen ekspresyon verilerinin kümelenmesi ile bu bilgilere ulaşılmaya çalışılmakta ve genler hakkında bilgi elde edinilmesini sağlamaktadır.

4.1. Kümeleme Problemi

n-boyutlu m noktadan oluşan R^n uzayında sonlu sayıda elemana sahip A kümesini ele alalım: $A=\{a^1,a^2,...,a^m\}$, $a^i \in R^n$, $i=1,...,m$.

Kümeleme probleminin amacı, A kümesindeki noktaları, verilen *k* adet ayrık A^j, $j=1,...,k$ alt kümeye, önceden tanımlanmış aşağıdaki kurallara göre ayırmaktır:

1. $A^j \neq \phi,\ j=1,...,k;$
2. $A^j \cap A^l = \phi,\ j,l=1,...,k,\ j \neq l;$
3. $A=\cup_{j=1}^{k} A^j;$

$A^j,\ j=1,\dots,k$ alt kümelerine, küme (cluster) adı verilir. Yukarıda verilen kuralları kısaca açıklamak gerekirse; 1'de oluşacak her bir kümenin boş küme olmaması; 2'de hiçbir küme çiftinin ortak bir elemana sahip olmaması; 3'de, kümelerin birleşiminin veri kümesine eşit olması anlamına gelir (Xu ve Wunsch, 2009).

Aynı kümeden olan noktalar birbirlerine benzer ve farklı kümeden olan noktalar ise birbirlerinden farklıdırlar. Veri noktalarının benzerlikleri benzerlik ölçüsü (similarity measure) olarak adlandırılan bir ölçümle tanımlanır (Anderberg, 1973). Bu ölçüm incelenen noktanın ait olduğu kümenin merkezine uzaklığı ile tanımlanır. Benzerlik ölçütü olarak $x_1, x_2 \in \mathbb{R}^n$ için $d(x_1, x_2)$ gibi bir uzaklık fonksiyonu kullanılabilir. Burada $d(x_1, x_2)$ fonksiyonunun seçimi uygulama bağımlıdır.

4.1.1. Mesafe ve Benzerlik Ölçüleri

En sık kullanılan mesafe ölçüm aracı Öklid mesafesidir. İki boyutlu X kümesi üzerinde $x=(x_1,x_2)$ ve $y=(y_1,y_2)$ elamanları için $d(x,y)$ öklid uzaklığı aşağıdaki gibi hesaplanır:

$$d(x,y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$$

Öklid mesafesi aşağıdaki özellikleri sağladığı için bir metriktir:

1. Simetriklik

$$d(x,y)=d(y,x)$$

2. Pozitiflik

$\forall x,y \in X$ için $d(x,y)\geq 0$

3. Üçgen eşitsizliği

$\forall x,y \in X$ için $d(x,y)\leq d(x,z)+d(z,y)$

4. Dönüşümlülük

$\forall x,y \in X$ için $d(x,y)=0$ ise $x=y$

Veri setlerinde bazı özelliklerin birimi ve değer aralıkları diğerlerinden çok farklı olduğunda bu özellikler baskınlık gösterebilmektedir. Bu durumda bu veriler üzerinde Öklid uzaklığını kullanmak anlamlı sonuçlar üretmeyecektir. Bu yüzden bu verilerin normalize edilmesi ve normalize edilmiş veriler üzerinde Öklid uzaklığı hesaplanmalıdır. Bunun için istatistikte en sık kullanılan metod z-skorlarıdır.

x_{il} z-skoru x_{il}^* özelliğin değeri, s_l özelliğin standart sapması, m_l özelliğin ortalaması olmak üzere z-skorları aşağıdaki gibi hesaplanmaktadır:

$$x_{il}=\frac{x_{il}^*-m_l}{s_l},\ i=1,...,N,\ l=1,...,d\ ,$$

$$m_l = \frac{1}{N}\sum_{i=1}^{N} x_{il}^{*},$$

$$s_l = \sqrt{\frac{1}{N-1}\sum_{I=1}^{N}(x_{il}^{*} - m_l)^2}$$

Diğer bir normalizasyon yöntemi maksimum ve minimum değerleri baz alan yaklaşımdır. Bu yöntem ile değerler [0,1] aralığına indirgenmektedir,

$$x_{il} = \frac{x_{il}^{*} - \min(x_{il}^{*})}{\max(x_{il}^{*}) - \min(x_{il}^{*})}$$

Öklid uzaklığının özel bir durumu aşağıdaki formül ile ifade edilen Minkowski uzaklığıdır:

$$d(x,y) = \left(\sum_{l=1}^{d} \sqrt[p]{\left|x_{il} - x_{jl}\right|}\right)^{p}$$

Eğer *p=2* ise formülün Öklid uzaklığını ifade edeceği açıktır. Burada p=1 alırsak aşağıdaki Manhattan uzaklığını elde ederiz:

$$d(x,y) = \sum_{l=1}^{d} \left|x_{il} - x_{jl}\right|$$

Diğer bir benzerlik ölçeği kosinüs uzaklığıdır. Kosinüs uzaklığı iç çarpımın normalleştirilmiş halidir. Burada değer ne kadar büyük çıkarsa karşılaştırılan nesneler birbirine o kadar benzerdir yani özellik uzayında nesnelerin paralellik gösterdiği anlamına gelir. Kosinüs benzerliği aşağıdaki formülle ifade edilir:

$$S(x_i, x_j) = \cos\alpha = \frac{x_i\, x_j}{\|x_i\|\|x_j\|}$$

Kosinüs benzerliği özellikle doküman kümeleme uygulamalarında oldukça kullanışlı bir ölçektir.

4.1.2. Kümeleme Probleminin Matematiksel Modellenmesi

En yaygın olarak incelenen ve kullanılan kümeleme yöntemleri bölümleme tabanlı kümeleme kategorisine girer. Bu yöntemlerde, kümeleme tanımıyla örtüşen bir amaç fonksiyonunu optimize edecek şekilde tek bir bölümleme elde edilmeye çalışılmaktadır. Yerel optimuma ulaşmak için bir başlangıç çözümünden

iteratif olarak iyileştirme stratejisi izlenir. Bölümleme tabanlı kümeleme, en doğal kümeleme yaklaşımıdır denilebilir.

$d(x, y)$: x ve y noktaları arasındaki uzaklık olsun. Bu durumda kümeleme problemi aşağıdaki optimizasyon problemine dönüşür:

$$\Psi(x, w) = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} w_{ij}.d(x^j, a^i) \to \min \quad (1)$$

$$x = (x^1, ..., x^k) \in \mathbb{R}^{nxk}, \quad (2)$$

$$\sum_{j=1}^{k} w_{ij} = 1, \ \text{i=1,...,m}, \quad (3)$$

$$w_{ij} = 0 \text{ veya } 1, \ \ i = 1, \dots, m, \ \ j = 1, \dots, k. \quad (4)$$

Burada *w*, *mxk* boyutlu matris olmak üzere w_{ij}: a^i örneğinin j kümesine aşağıdaki koşullara göre ait olma ağırlığıdır;

$$w_{ij} = \begin{cases} 1, \text{ eğer } a^i \text{ elemanı } j \text{ kümesine atandıysa,} \\ 0, \text{ aksi halde.} \end{cases}$$

(1)-(4) modeli kümeleme probleminin karma tam sayılı doğrusal olmayan programlama modelidir (Bagirov, 2008). (1)-(4) deki kümeleme probleminin Pürüzlü Dış bükey olmayan (Nonsmooth Nonconvex) formülasyonu ise aşağıdaki şekilde verilebilir (Xu ve Wunsch, 2009):

$$f_k(x^1, ..., x^k) = \frac{1}{m}\sum_{i=1}^{m} \min_{j=1,k} \left\{ d(x^j, a^i) \right\} \to \min \quad (5)$$

$$(x^1, \dots, x^k) \epsilon \mathbb{R}^{n \times k}, \quad (6)$$

Hem ψ_k hem de f_k fonksiyonlarına küme (cluster) fonksiyonları denir.

Literatürde yer alan kümeleme problemlerinin genel anlamda matematiksel modeli yukarıda açıklanmıştır. Fakat yoğunluk tabanlı yöntemler için matematiksel model olmadığı görülmüştür. Bu hususta yoğunluk tabanlı yöntemler için matematiksel model geliştirilmiş ve literatüre katkı yapılmıştır (Tanir ve ark., 2017). Geliştirilen model aşağıda ifade edilmiştir:

$$u_k(w) = \frac{1}{m} \sum_{\substack{i,l=1 \\ i \neq l}}^{m} \min_{j=1,k} \left\{ w_{lj}.w_{ij}.d(a^l, a^i) \right\} \longrightarrow \min \quad (7)$$

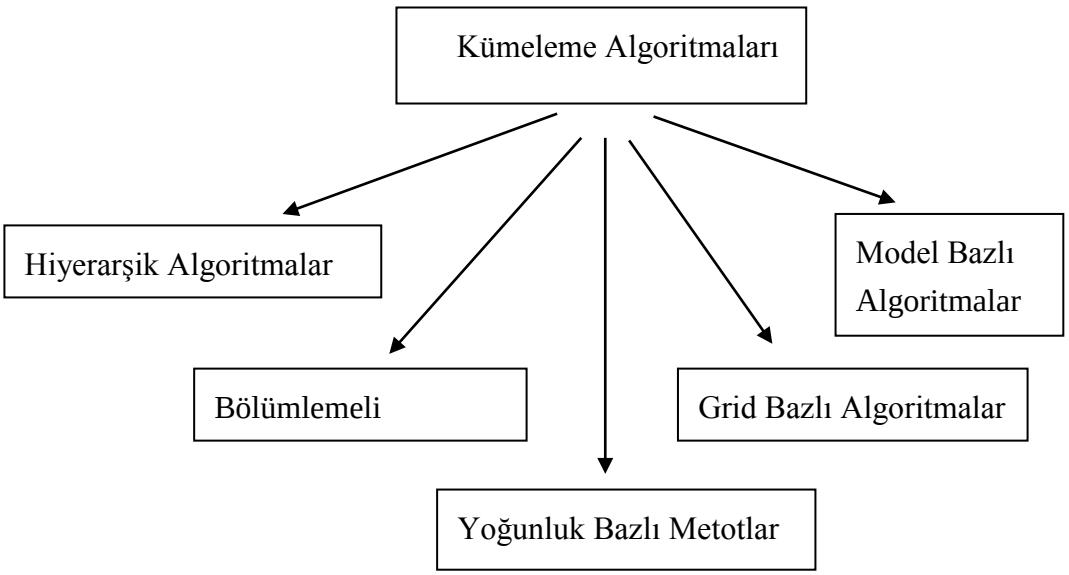
$$\sum_{j=1}^{k} w_{ij} = 1, \ \text{i=1,...,m}, \quad (8)$$

$$w_{ij} = 0 \text{ veya } 1, \ \ i = 1, \ldots, m, \ \ j = 1, \ldots, k \quad (9)$$

Burada (7) koşulu ayrık k kümenin aynı kümede bulunan herhangi iki elemanının birbirine en yakın mesafede olması, (8) koşulu her bir elemanın yalnız bir kümeye ait olması ve (9) koşulunda eğer a^i elemanı j kümesine atandıysa w_{ij}'nin 1 değerini aksi halde 0 değerini alması anlamına gelir. (7)–(9) modeli tamsayılı doğrusal olmayan 0-1 matematiksel programlama modelidir.

4.2. Kümeleme Algoritmaları

Değişik kaynaklarda kümeleme algoritmaları farklı biçimlerde sınıflandırılmaktadır. Kümeleme algoritmalarının sınıflandırılmasında çok değişkenli veri analizi kitaplarında kullanılan en genel ayrım hiyerarşik ve hiyerarşik olmayan kümeleme metotları ayrımıdır. Çeşitli kümeleme metotlarının yer aldığı çeşitli sınıflamalar yapılabilir. Kümeleme metotlarının sınıflandırılmasına ilişkin bir sınıflama örneği Şekil 4.2' deki gibidir.



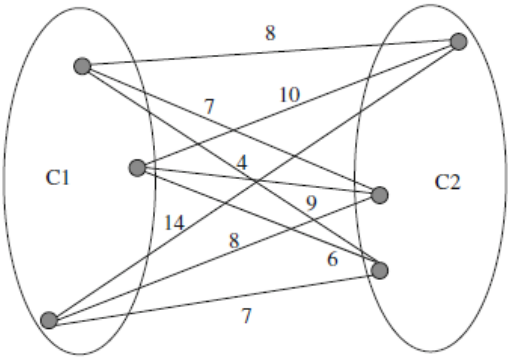
Şekil 4.2. Sınıflandırılmış Kümeleme Metotları

4.2.1. Hiyerarşik Kümeleme Algoritmaları

Hiyerarşik kümeleme algoritmaları basitliği ve orta boyuttaki veri hacimleri için kabul edilebilir performansa sahip olmasından dolayı en sık kullanılan kümeleme algoritmalarıdır. Birleştirici ve ayrıcı olmak üzere iki sınıfa ayrılmaktadır. Birleştirici kümelemede başlangıçta her nokta başlangıç küme merkezleri olarak düşünülür. Birbirine en yakın iki küme her bir adımda birleştirilerek yeni bir küme olarak alınır. Bu adımlar tek bir küme elde edilinceye kadar devam eder. C_i *ve* C_j iki küme olmak üzere aralarındaki mesafe aşağıdaki yöntemler ile belirlenir:

- Tekil Bağlantı (Single Link): $d(C_i, C_j)$ mesafesi iki küme içerisinde birbirine en yakın iki $x \in C_i$ ve $y \in C_j$ elemanları arasındaki mesafedir.
- Tam Bağlantı (Complete Link): $d(C_i, C_j)$ mesafesi iki küme içerisinde birbirine en uzak iki $x \in C_i$ ve $y \in C_j$ elemanları arasındaki mesafedir.
- Ortalama Bağlantı (Avarage Link): $d(C_i, C_j)$ mesafesi iki küme arasındaki tüm elemanların ikili mesafelerinin ortalama mesafesidir.

Şekil 4.3'de bu yöntemlerin uygulanışı verilmiştir. Ayırıcı algoritmada ise başlangıç tüm kümedir ve sonraki adımlarda küme daha küçük parçalara ayrılır. Birleştirci kümelemede çıktı ağaç şeklinde dendogramdır ve bu dendogram yatay bir çizgiyle bölünerek istenilen sayısında küme elde edilir.



Şekil 4.3. Hiyerarşik kümeleme örneği. Tekil bağlantı 4 birim, tam bağlantı 14 birim ve ortalama bağlantı 8,11 birimdir.

Birleştirici kümeleme algoritmasında iki küme arasındaki en kısa mesafeyi bulmak $O(n^2)$ karmaşıklığa sahiptir. Bu işlem *n-1* kez tekrar etmektedir bu yüzden algoritmanın karmaşıklığı $O(n^3)$ olmaktadır. İyi bir veri yapısı kullanılarak algoritmanın karmaşıklığı $O(n^2 \log n)$ zaman karmaşıklığına düşürülebilmektedir.

4.2.2. Yoğunluk Tabanlı Kümeleme Algoritmaları

Yoğunluk tabanlı kümeleme yöntemlerinin öne çıkan özelliği, farklı ve düzensiz şekillerdeki kümeleri belirleyebilmesidir. Bir kümeyi oluşturmak için, birbirine bağlantılı yoğun noktalar belirlenir. Yani bir küme, yoğunluğun yönlendirdiği herhangi bir yönde genişleyebilir. Böylece herhangi bir şekildeki küme oluşturulabilir. Yoğunluk değerlendirmesinin doğal bir etkisi olarak, yoğunluk tabanlı kümeleme yöntemleri gürültü (sıradışı veya anormal özelliklere sahip veri) ile baş edebilirler. Ayrıca, bu yaklaşımı benimseyen yöntemler genellikle düşük hesaplama karmaşıklığına sahiptirler. En belirgin dezavantaj ise, ortaya çıkan kümelerin nasıl değerlendirileceği ve yorumlanacağıdır.

Yoğunluk-tabanlı kümelemenin temsilcisi DBSCAN (Density Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points to Identify the Clustering Structure), DENCLUE (Density-based Clustering) algoritmaları kabul edilir (Jain ve Dubes, 1988) . İsimlerinden de anlaşılacağı gibi bu algoritmalar veri yoğunluğunu temel alan yöntemlerdir. Bu algoritmaların temel prensibi bir kümedeki her veri noktası için, belirli bir yarıçap (eps) içinde en az (minpts) sayıda veri noktası içeriyor olmasıdır. DENCLUE algoritması diğer yoğunluğa dayalı kümeleme algoritmalarının genelleştirilmiş halidir.

4.2.3. Bölümlemeli Algoritmalar

Bölümlemeli algoritmalar bazı keyfi başlangıç merkezleri belirleyerek direk olarak kümeleme yapmayı amaçlar. Bu algoritmalar için en sık kullanılan algoritma k-ortalamalar algoritmasıdır.

4.2.3.1. K-ortalamalar Algoritması

K-ortalamalar algoritması uygulaması ve anlaşılması kolay bir algoritmadır. Algoritmada *k* küme sayısı belirlenir ve kümelerin merkezleri bulunmaya çalışılır. Bunun için başlangıçta *k* tane rasgele nokta belirlenir, bu noktalara başlangıç merkezleri de denir.

K-ortalamalar kümeleme algoritmasının genel matematiksel modeli aşağıdaki gibi verilebilir.

n-boyutlu *m* noktadan oluşan R^n uzayında sonlu sayıda elemana sahip *A* kümesini, bu kümenin *k* tane ayrık alt kümesinin merkezlerine sahip *C* kümesini ele alalım:

$A=\{a^1,a^2,...,a^m\}$, $a^i \in R^n$, $i=1,...,m$, $S_j=\{a \mid a,\ j \text{ nolu kümenin elemanı}\}$ $j=1,...,k$, $C=\{c^j \mid j=1,...,k\}$ ve $d(x,y)$: x ve y noktaları arasındaki uzaklık olmak üzere algoritmanın amacı aşağıda verilen amaç fonksiyonunu minimize etmektir.

$$\psi(x,w)=\min\left\{\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k}w_{ij}.d(c^j,a^i)\right\}$$

Burada: $\sum_{j=1}^{k} w_{ij}=1,\ i=1,...,m$,

$$w_{ij}=0 \text{ veya } 1,\ i=1,...,m,\ j=1,...,k$$

ve w_{ij} : a^i noktasının *j* kümesine ait olma durumudur. Standart olarak w_{ij} değeri aşağıdaki şekilde belirlenir:

$$w_{ij}=\begin{cases}1, \text{ eğer a}^{\text{i}} \text{ j nolu kümeye ait ise,}\\ 0, \text{ aksi durumda}\end{cases}$$

K-means algoritmasının adımları aşağıdaki gibidir:

- Rasgele k tane başlangıç merkezi belirle
- Tekrar
  - Her noktayı kendisine en yakın olan merkezin ait olduğu kümeye ata
- Her kümenin merkezini tekrar hesapla:

  $c^k=\frac{\sum_{a^i \in S_k} a^i}{|S_k|}$, $|S_k|$ k. kümenin eleman sayısı

- Merkezlerin değişimi belirli bir ε değerinden küçük oluncaya kadar tekrar edilir (Kim ve ark., 2006) :

  $$\left|\frac{c^t-c^{t-1}}{c^t}\right|<\varepsilon,\ t=1,...,k$$

Birinci adımda k tane rasgele nokta belirlenir. İkinci adımda noktalar kendisine en yakın olan merkezin bulunduğu kümeye atanır ve küme merkezleri güncellenir. Genellikle uzaklık ölçmek için Öklid uzaklığından yararlanılır. Üçüncü adımda noktalar güncellenmiş merkezlere göre kümelere atanır ve merkezler yeni kümelere göre tekrar güncellenir. Merkezlerin değişimi belirli bir değerinden küçük oluncaya kadar ikinci ve üçüncü adımlar tekrar edilir.

K-ortalamalar algoritması ψ ile verilen amaç fonksiyonunu minimize etmeye çalışır ve bunun için de her iterasyonda kümelerin merkezleri güncellenir. Sonuç olarak algoritma genellikle lokal minimuma yakınsar. Global minimumun bulunması ise NP-tam problemlerdendir. Burada iterasyon sayısına l, nokta sayısına m ve küme sayısına k dersek k-ortalamalar algoritmasının karmaşıklığı $O(lmk)$ olur.

4.3. K-ortalamalar Algoritması İçin Önerilen Başlangıç Merkezleri Belirleme Yöntemleri

K-ortalamalar algoritmasında başlangıç noktalarının seçimi çok önemlidir. Çünkü seçilen bu noktalar iterasyonun sonunda elde edilen final kümenin yapısında direk etkiye sahiptir. Ayrıca bu aşama algoritmanın performansında da etkiye sahiptir.

Khan ve ark. başlangıç noktalarını final kümenin başlangıç noktalarına oldukça yakın seçen Cluster Center Initialization Algorithm (CCIA) önermiştir (Khan ve Ahmad, 2004). Erisoglu ve ark. ise başlangıç noktalarını birbirlerine uzak seçmiş ve standart K-ortalamalar algoritmasına göre daha iyi kümeleme yaptığını belirtmiştir (Erisoglu ve ark., 2011). Başlangıç noktalarının rasgele seçilmesi K-means algoritmasının genel olarak uygun olmayan lokal optimumlara takılmasına sebep olmaktadır. Bu yüzden daha iyi sonuçlara ulaşabilmek için K-means algoritmasının birçok kere çalıştırılması gerekmektedir. Ayrıca başlangıç noktalarının rasgele seçilmesi her defasında farklı kümeleme sonucu üretecektir. Bu yüzden aynı başlangıç noktalarını üretecek bir algoritma geliştirilmesi K-ortalamalar algoritmasının farklı sonuçlar üretmesinin önüne geçecektir.

Bu tez çalışmasında K-ortalamalar algoritmasında başlangıç noktaları üreten algoritmalar önerilmiş ve veri setleri üzerinde standart K-ortalamalar algoritması ile karşılaştırma yapılmıştır.

4.3.1. Önerilen Yöntem - 1

İteratif kümeleme algoritmalarında başlangıç noktası seçme prosedürü çözümün iyileştirilmesinde önemli bir etkiye sahiptir. Önerilen yöntemde başlangıç noktaları birbirinden oldukça uzak seçilmesine dayanır. İlk olarak noktalar arasından birbirine en uzak iki nokta seçilir. Burada uzaklık ölçümü olarak öklid uzaklığı kullanılır.

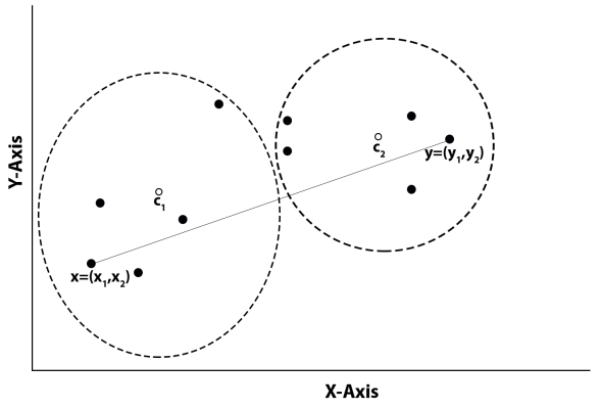
İkinci adımda seçilen bu noktaları merkez kabul ederek noktaların merkezlere yakınlığına göre iki küme elde edilir. Daha sonra iki kümenin merkez noktaları güncellenir. Şekil 4.4'de bu noktalar c_1 ve c_2 olarak elde edilmiştir. Elde edilen merkezler başlangıç noktası olarak I kümesine eklenir. Üç kümeye ayırmak gerekiyorsa seçilen bu iki noktaya en uzak nokta seçilir bulunan nokta c_3 noktası olarak I kümesine eklenir:

$$d = \sum_{i=1}^{m} d(a^i, c_1) + d(a^i, c_2)$$

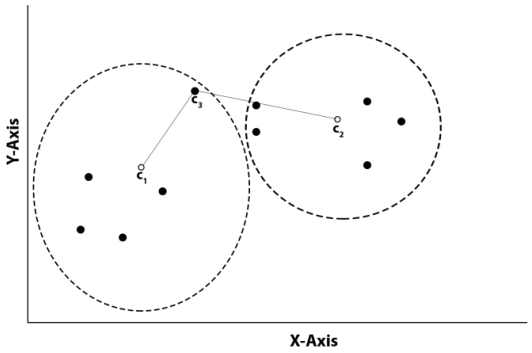
Dört kümeye ayırmak gerekiyorsa seçilen üç noktaya en uzak nokta seçilir ve dördüncü nokta olarak başlangıç noktalarına eklenir:

$$d = \sum_{i=1}^{m} d(a^i, c_1) + d(a^i, c_2) + d(a^i, c_3)$$

Başlangıçta verilen küme sayısı elde edilinceye kadar bu işlem devam ettirilir.

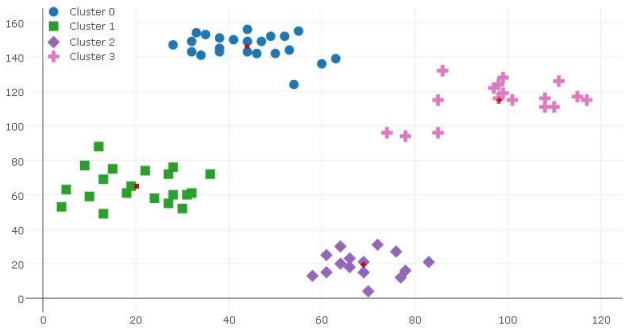


Şekil 4.4. 10 tane noktanın iki kümeye birbirine en uzak seçilen iki merkez noktasına göre parçalanışı. Düz çizgi iki nokta arasındaki mesafeyi göstermektedir.

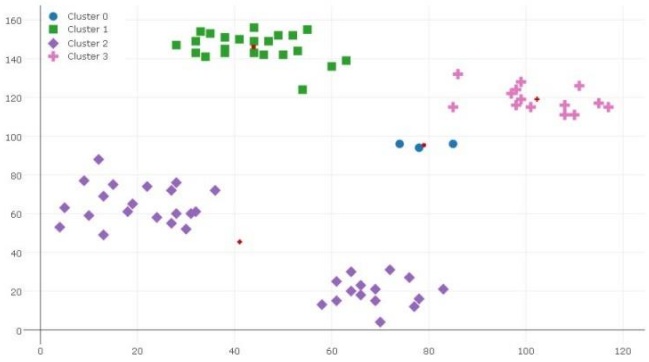


Şekil 4.5. c_1 ve c_2 noktalarına en uzak üçüncü nokta c_3 seçilir. Düz çizgi iki nokta arasındaki mesafeyi göstermektedir.

Örnek olarak yöntemin başarısını ölçmek için gerçek hayat verilerinin yer aldığı UCI veritabanında (http://archive.ics.uci.edu/ml/datasets.html) Ruspini veri seti için önerilen yöntemi kullanan ve random yöntemi kullanan K-means algoritmasının sonuçları aşağıdaki çizelge ve grafiklerde gösterilmiştir. Random yöntemi kullanan K-ortalamalar algoritmasının Ruspini veri setinde 10 defa çalıştırılmasından elde edilen hata oranı 13.69 çıkarken önerilen yöntemin hata oranı 0 çıkmıştır.



Şekil 4.6. Ruspini veri setinde önerilen metodu kullanan K- ortalamalar Algoritması sonucu



Şekil 4.7. Ruspini veri setinde rastgele başlangıç merkezleri üreten K-ortalamalar Algoritması sonucu

Çizelge 4.1. Ruspini Veri Seti Üzerinde Önerilen Yöntemi Kullanan K-ortalamalar Algoritmasının Sonucu

Kümeler	Kümeleme Sonrası Kümelerdeki Nokta Sayısı	Hangi Kümeden Geldiği			
		1	2	3	4
C1	23	23	0	0	0
C2	20	0	20	0	0
C3	17	0	0	17	0
C4	15	0	0	0	15

4.3.2. Önerilen Yöntem – 2

Önerilen ikinci yöntem de başlangıç noktalarının birbirinden uzak ve yoğun kümelerden seçilmesine dayanır. Önerilen yöntem - 1'de olduğu gibi birinci adımda noktalar arasından birbirine en uzak iki nokta seçilir, ikinci adımda seçilen noktalar merkez kabul edilerek noktaların merkezlere yakınlığına göre iki küme elde edilir ve iki kümenin merkez c_1 ve c_2 noktaları olarak güncellenir ve bu noktalar başlangıç noktaları olarak I kümesine eklenir. Üçüncü adımda bu iki kümeden en fazla noktaya sahip olan kümenin (varsayalım ki $|S_2| > |S_1|$ olsun) birbirine en uzak iki noktası bulunur ve bu küme için o iki nokta merkez kabul ederek kümeleme yapılır. Böylece S_2 kümesi S_{21} ve S_{22} olmak üzere iki kümeye ayrılır ve c_2 noktası I kümesinden çıkarılıp c_{21} ve c_{22} noktaları I kümesine eklenir. Başlangıçta belirlenen k sayısına ulaşıncaya kadar I kümesine aynı işlemleri tekrar ederek noktalar eklenmeye devam edilir. Son olarak belirlenen k tane başlangıç noktasına göre k-ortalamalar algoritması uygulanır.

4.3.3. Önerilen Yöntem - 3

Bu yöntemde Erisoglu ve arkadaşlarının (Erisoglu ve ark., 2011) önerdiği yöntemden yola çıkarak yeni bir başlangıç merkezleri belirleme yöntemi

önerilmiştir. Yöntemde verilen küme sayısına göre birbirine en uzak noktaların bulunması amaçlanmıştır.

Birbirine en uzak merkez noktalar aşağıdaki şekilde belirlenir:

Adım 1. Birbirine en uzak 2 nokta (c_1, c_2) bulunur ve I kümesine eklenir.

Adım 2. Bulunan 2 noktaya toplam mesafesi en uzak 3. nokta (c_3) bulunur ve I kümesine eklenir. Toplam mesafe aşağıdaki gibi hesaplanır:

$$d = \sum_{i=1}^{m} d(c_i, c_1) + d(c_i, c_2)$$

Adım 3. Bulunan 3 noktaya toplam mesafesi en uzak 4. nokta (c_4) bulunur ve I kümesine eklenir. Toplam mesafe aşağıdaki gibi hesaplanır:

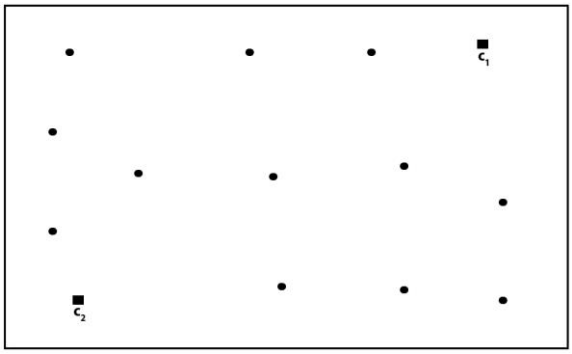
$$d = \sum_{i=1}^{m} d(c_i, c_1) + d(c_i, c_2) + d(c_i, c_3)$$

Adım 4. I kümesindeki noktaların merkez noktası hesaplanır.

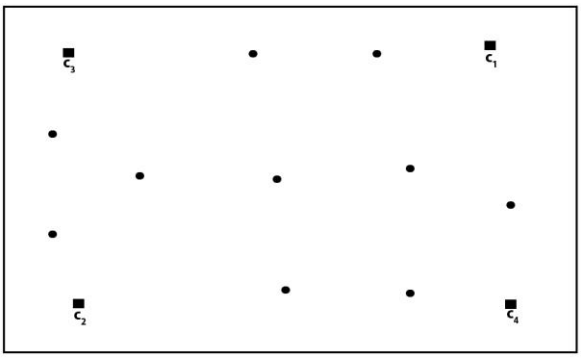
$$m = \frac{\sum_{c_i \in I} c_i}{|I|}$$

Adım 5. Birbirine en uzak mesafede olan 2 komşu nokta ile merkez (m) noktaya en uzak mesafedeki farklı bir nokta bulunur ve I kümesine eklenir. Buradaki komşu noktalardan kastedilen I kümesindeki noktaların birleştirilmesiyle elde edilebilecek bir çokgende bir doğru parçasıyla bir birine bağlanan noktalardır. Örneğin Şekil 17'de c_1 noktası c_3 ve c_4, c_2 noktası c_3 ve c_4 ile komşudur.

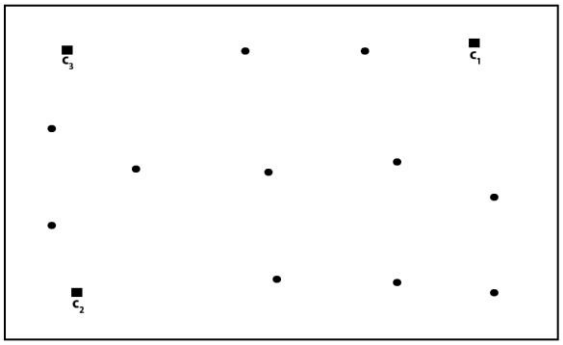
Adım 6. I kümesinin eleman sayısı başlangıçta belirlenen k sayısına ulaşıncaya kadar Adım 5 tekrarlanır.



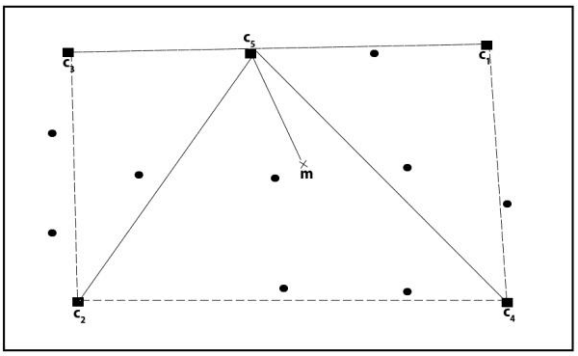
Şekil 4.8. Birbirine en uzak 2 nokta bulunur



Şekil 4.9. Bulunan 2 noktaya toplam mesafesi en uzak 3. nokta bulunur



Şekil 4.10. Bulunan 3 noktaya toplam mesafesi en uzak 4. nokta bulunur



Şekil 4.11. Birbirine en uzak mesafede olan 2 komşu nokta ile merkez (m) noktaya en uzak mesafedeki farklı bir nokta bulunur

4.4. Önerilen Yöntemlerin Karşılaştırılması

Önerilen yöntem Intel Corei7 4700HQ 2.4 GHz CPU, 32GB of RAM ve Windows 8.1 64-bit işletim sistemi üzerinde Python dilinde yazılmıştır. Başarısını ölçmek için gerçek hayat verilerinin yer aldığı UCI (http://archive.ics.uci.edu/ml/datasets.html) Machine Learning Repository'den 4 tane veri seti seçilmiştir. Seçilen veri setleri üzerinde standart K-ortalamalar ve önerilen K-ortalamalar algoritmasının hesaplama sonuçları karşılaştırılmıştır. Burada karşılaştırma kriteri olarak "Hata oranı" ve "Rand indeks" kullanılmıştır. Algoritmanın ε yakınsamasında parametresi 0,2 olarak seçilmiştir.

Standart K-ortalamalar rasgele başlangıç noktası belirlediğinden her defasında farklı sonuç üretmektedir. Bu yüzden algoritmanın performansını ölçmek için 10 defa çalıştırılıp ortalama değeri hesaplanmıştır.

Hata oranı aşağıdaki eşitlik ile hesaplanır:

$$\text{Hata Oranı} = \frac{\text{Yanlış sınıflanan nesne sayısı}}{\text{Toplam nesne sayısı}}$$

Rand indeks istatistikte ve kümelemede kümeler arası benzerlik ölçümünde sıklıkla kullanılan bir kriterdir. $A = \{a^1, a^2, ..., a^m\}$ kümesinin $P = \{p^1, p^2, ..., p^k\}$ ve $Q = \{q^1, q^2, ..., q^k\}$ k tane alt küme içeren iki parçalanışı olmak üzere her bir $\{a^i, a^j\}$ ikilisinin 4 farklı durumu vardır:

a: a^i *ve* a^j noktaları P ve Q da aynı alt kümede yer almaktadır.

b: a^i *ve* a^j noktaları P'de farklı fakat Q'da aynı alt kümede yer almaktadır.

c: a^i *ve* a^j noktaları P'de aynı fakat Q'da farklı alt kümede yer almaktadır.

d: a^i *ve* a^j noktaları P'de ve Q'da farklı alt kümede yer almaktadır.

Buna göre rand indeks aşağıdaki eşitlik ile hesaplanır:

$$Rand = \frac{a+d}{a+b+c+d}$$

Karşılaştırmada kullanılan veri setlerinin açıklamaları aşağıdaki gibidir:

Iris Veri Seti : Bu veri seti 150 örnek içermekte ve her bir örnek 4 özellik (çanak yaprak uzunluğu, çanak yaprak genişliği, taç yaprak uzunluğu ve taç yaprak genişliği) barındırıp 3 sınıftan (Iris setosa, Iris versicolor, ve Iris virginica) oluşmaktadır. Küçük ve az sayıda özellik barındırmasından dolayı kümeleme çalışmalarında sıklıkla çalışılan veri setidir.

Çizelge 4.2. Iris veri seti karşılaştırma sonuçları

Metotlar	Hata Oranı	Rand İndeks
Rasgele başlangıç üreten yöntemi kullanan K-ortalamalar	13.83	0.8639
Önerilen metot - 1'i kullanan K-ortalamalar	**10.00**	**0.8859**
Önerilen metot – 2'yi kullanan K-ortalamalar	11.33	0.8859
Erisoglu ve arkadaşlarının metodu	10.7	0.8797

2. Wine Veri Seti : Bu veri seti 178 örnek ve 13 özellik içermekte (Alcohol, Malic, Ash, Alcalinity, Magnesium, Phenols, Flavanoids, Non-Flavanoids, Proanthocyanins, Color, Hue, (1OD280/OD315) of diluted wines ve Proline) ve 3 sınıftan oluşmaktadır. Bu veri setindeki özelliklerin değerler aralığı çok geniş olduğundan verilerin aşağıdaki formül kullanılarak z-skorları hesaplanmış ve normalleştirilmiştir.

Çizelge 4.3. Iris veri seti karşılaştırma sonuçları

Metotlar	Hata Oranı	Rand İndeks
Rasgele başlangıç üreten yöntemi kullanan K-ortalamalar	5.0561	0.9330
Önerilen metot - 1'i kullanan K-ortalamalar	3.9325	0.9482
Önerilen metot – 2'yi kullanan K-ortalamalar	**3.3707**	**0.9542**
Erisoglu ve arkadaşlarının metodu	3.40	0.9543

3. The Glass Veri Seti : Bu veri seti 214 örnekten oluşmakta ve her bir örnek 9 özellik barındırmaktadır. Veri seti 7 cam tipi içermektedir fakat 4 nolu id veri barındırmamaktadır. Bu yüzden altı kümeden oluşmaktadır.

Çizelge 4.4. The Glass veri seti karşılaştırma sonuçları

Metotlar	Hata Oranı	Rand İndeks
Rasgele başlangıç üreten yöntemi kullanan K-ortalamalar	15.8878	0.9218
Önerilen metot - 1'i kullanan K-ortalamalar	**11.2149**	**0.9683**
Önerilen metot – 2'yi kullanan K-ortalamalar	13.0841	0.9494
Erisoglu ve arkadaşlarının metodu	-	-

4. The Ruspini Veri Seti: Bu veri seti 75 örnekten oluşmakta ve her bir örnek iki özellik barındırıp 4 sınıftan oluşmaktadır.

Çizelge 4.5. The Ruspini veri seti karşılaştırma sonuçları

Metotlar	Hata Oranı	Rand İndeks
Rasgele başlangıç üreten yöntemi kullanan K-ortalamalar	21.8667	0.8887
Önerilen metot - 1'i kullanan K-ortalamalar	0	1
Önerilen metot – 2'yi kullanan K-ortalamalar	0	1
Erisoglu ve arkadaşlarının metodu	0	1

4.5. Önerilen Yöntemlerin Genomik Verileri Kümeleme Sonuçları

Kümelemenin başarısını ölçmek için gerçek hayat gen ekspresyon verileri kullanılmıştır. Elde edilen sonuçlar farklı çalışmalardan elde edilen sonuçlar ile karşılaştırılmıştır. Yeast Sporulation, Human Fibroblasts Serum ve Rat CNS olmak üzere gerçek hayattan 3 gen ekspresyon veri seti alınmıştır. Bu verilere aşağıdaki adresten erişilebilir: http://anirbanmukhopadhyay.50webs.com/data.html

1) Yeast Sporulation : Bu veri seti 7 tane zaman noktasına (0, 0.5, 2, 7, ve 11.5 h) sahip olan 6118 genden oluşmaktadır. Bu değerler tomurcuklanan mayanın sporlanma aşamasında elde edilmiştir. 6118 gen arasında bazılarının ekspresyon seviyesi çok küçük değişimlere sahip olduğundan bu veriler analizden çıkarılmıştır. Sonuç olarak elde edilen veri setinde 474 gen yer almıştır.

2) Human Fibroblasts Serum : Bu veri seti 13 boyutlu 12 tane zaman noktasına (0, 0.25, 0.5, 1, 2, 4, 6, 8, 12, 16, 20 ve 24 h) sahip olan 8613 genden oluşmaktadır. Bu değerler insan genlerinden elde edilmiştir. 8613 gen arasında bazılarının ekspresyon seviyesi çok küçük değişimlere sahip olduğundan bu veriler analizden çıkarılmıştır. Sonuç olarak elde edilen veri setinde 517 gen yer almıştır.

3) Rat CNS: Bu veri seti 9 tane zaman noktasına sahip olan 112 genden oluşmaktadır. Bu değerler farenin merkezi sinir sistemi gelişimi boyunca elde edilmiştir.

Standart K-ortalamalar rasgele başlangıç noktası belirlediğinden her defasında farklı sonuç üretmektedir. Bu yüzden algoritmanın performansını ölçmek için 10 defa çalıştırılıp ortalama değeri hesaplanmıştır. Algoritmanın yakınsamasında ε parametresi 0.2 olarak seçilmiştir. Sonuçlar karşılaştırılırken Silhouette İndeks değeri kullanılmıştır.

Silhouette İndeks kümelemenin kalitesini ölçmek için kullanılan doğruluk indeksidir. a değeri ait olduğu kümedeki tüm noktalara ortalama uzaklığı ve b değeri diğer kümelerdeki noktalara ortalama uzaklığının minimum değeri olmak üzere belirli bir noktanın Silhouette İndeks değeri aşağıdaki formül ile hesaplanır:

$$s = \frac{b - a}{\max\{a, b\}}$$

Sihouette Index s(C) değeri tüm noktalar için hesaplanan değerlerin ortalamasıdır ve -1 ile 1 arasında değer alır. Ne kadar yüksek değere ulaşılırsa o kadar iyi kümeleme yapıldığı anlamına gelir. Buradaki noktalar bizim çalışmamızda gen anlamına gelmektedir.

K-ortalamalar algoritmasında küme sayısının başlangıçta parametre olarak girilmesi gerekmektedir. Bandyopadhya ve arkadaşları (Bandyopadhyay ve ark., 2007) en yüksek s(C) değerine her bir veri seti için küme sayısını 6 alarak ulaştığını belirtmiştir. Bu çalışmada da her bir veri seti için küme sayısı 6 olarak alınmış ve elde edilen sonuçlar Çizelge 4.6'da Erisoglu ve arkadaşlarının önerdiği algoritma, Bandyopadhya ve arkadaşlarının önerdiği SiMM-TS (Bandyopadhyay ve ark., 2007), VGA (Maulik ve Bandyopadhyay, 2003) algoritmaları ve literatürdeki diğer algoritmalar (average linkage (Jain ve Dubes, 1988), SOM (Tamayo ve ark., 1999), CRC (Kim ve ark., 2006)) ile karşılaştırılmıştır.

Çizelge 4.6. Gen Ekspresyon Veri Setlerinde Algoritmaların Karşılaştırmalı s(C) Değerleri

Algoritmalar	**Rat CNS**	**Yeast**	**Serum**
Standart K-means	0.4651	0.5888	0.4016
Önerilen metot – 3'ü Kullanan K-means	**0.5278**	0.6670	**0.5016**
Erisoglu ve ark. metodu	0.5061	**0.6768**	0.4776
SiMM-TS	0.5147	0.6247	0.4289
VGA	0.4542	0.5703	0.3443
Average linkage	0.3684	0.5007	0.3092
SOM	0.4122	0.5845	0.3345
CRC	0.4423	0.5622	0.3227

Çizelge 4.6'daki karşılaştırma sonuçlarına göre önerilen metot tüm veri setleri için Erisoglu ve arkadaşlarının metodu dışındaki diğer metodlara göre daha iyi sonuç vermiştir. Erisoglu ve arkadaşlarının metodu ise sadece "Yeast" veri setinde kısmen daha iyi sonuç verirken diğer veri setlerinde önerilen metot daha iyidir.

5. GELİŞTİRİLEN YAZILIM VE KULLANIM PRENSİPLERİ

Bu bölümde Ege Üniversitesi Moleküler Biyoloji Laboratuvarındaki DNA analiz cihazının ürettiği verileri işlemek için geliştirilen yazılım ve kullanım prensipleri anlatılmıştır.

Prof. Dr. Afig Berdeli'nin yöneticiliğini yaptığı bu laboratuvarda DNA analiz cihazının ürettiği ABIF formatındaki verilerin okunması ve sonraki aşamada verilerin analizi ile dizinler arasındaki mutasyonların belirlenmesi işlemi geliştirilen yazılımın ana problemidir. Yapılan analiz genel olarak iki dizin arasındaki farklılıkların belirlenmesi olduğundan yazılımda global ve lokal hizalama algoritmaları kullanılmıştır. Bu alandaki algoritmalardan olan Needleman–Wunsch, Smith Waterman dinamik algoritmaları ve indekslemeye dayalı arama yöntemlerinden sonek ağacı yöntemini kullanan MUMmer ve NUCmer algoritmaları tercih edilmiştir.

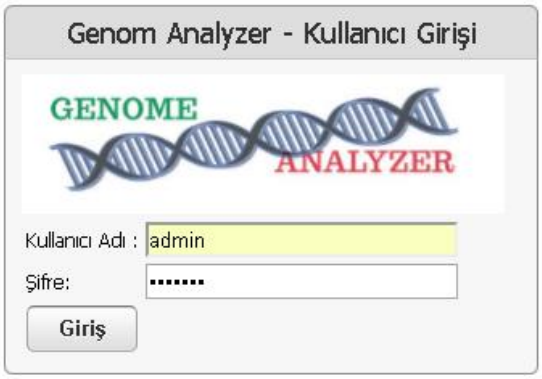
5.1. Genom Analyzer Web Yazılımı

Genom Analyzer genomik verilerin analizi için geliştirilen Microsoft .Net Framework platformunun 4.6.1 versiyonunda C# dilinde kodlanmış bir web uygulamasıdır. Bu programın alt yapısında Microsoft'un açık kaynak olarak geliştiricilere sunduğu .NET Bio Platformu kullanılmıştır (Acosta, 2011). Microsoft Biology Foundation (MBF) isimli bu açık kaynak kodlu yazılım .Net Framework kütüphanesi olarak tasarlanmıştır. Bu kütüphanenin amacı sık kullanılan FASTA, FASTAQ, GFF ve GENBANK gibi formatları açabilmek, formatlar arası dönüşüm yapabilmek, NCBI BLAST gibi web servis sağlayıcılarına bağlanıp bu servisleri kullanabilmek ayrıca DNA, RNA, protein dizilerini birbirleri ile karşılaştırabilmek, hizalamayabilmek ve bu dizileri işleyebilmek için genomik araştırma alanında yazılım geliştiren programcılara temel oluşturmaktır. MBF'yi geliştiriciler ücretsiz olarak https://github.com/dotnetbio/bio adresinden indirebilmektedir.

Genom Analyzer, .NET Bio Platformunun birçok fonksiyonunu alt yapısında barındırmakla birlikte ek olarak ABIF formatlı dosyaları da açabilme ve işleyebilme özelliğine sahiptir. Ayrıca yazılımda genomik araştırmacıların daha kolay analiz yapabilmelerini sağlamak için görsel grafikler yer almaktadır. Örneğin ABIF formatlı dosyaların kalitesini ölçebilmek için sekans pik

görüntüleri ve dizilerin istatistiklerini gösteren grafikler vardır. Bu yazılımın en temel amacı bu alanda geliştirilen yazılımların büyük çoğunluğunun masaüstü tabanlı olması ve Web'de de masaüstü programların sahip olduğu özelliklerin birçoğuna sahip bir portal geliştirmektir.

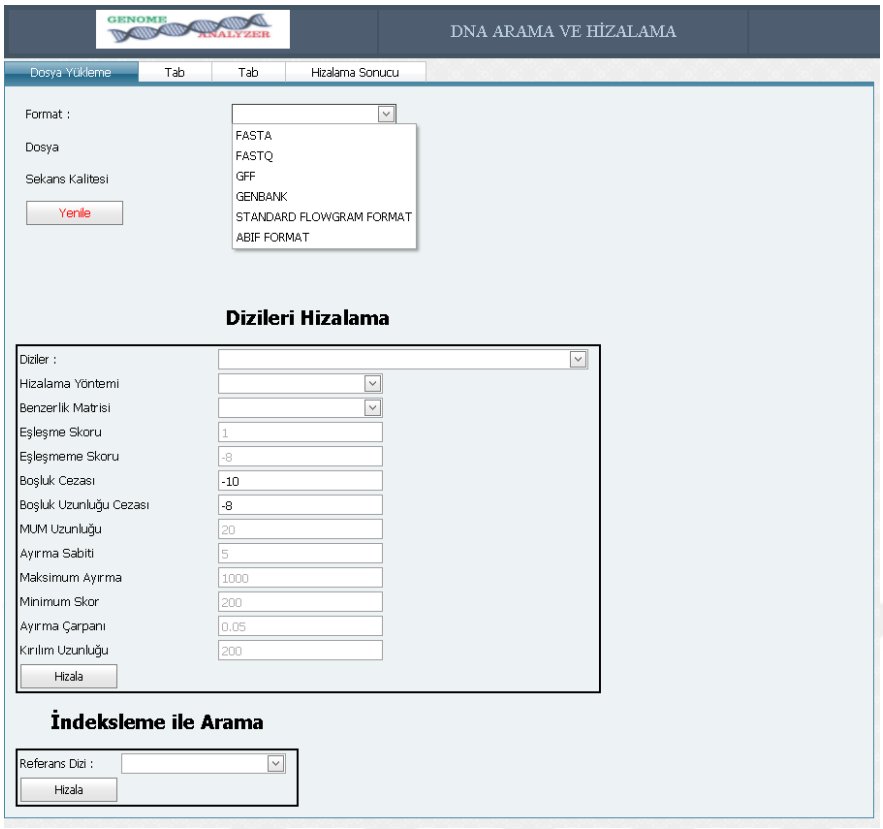
Program ilk açıldığında kullanıcı adı ve parola girişinin yapıldığı Şekil 5.7'deki kullanıcı girişi ekranı gelmektedir.



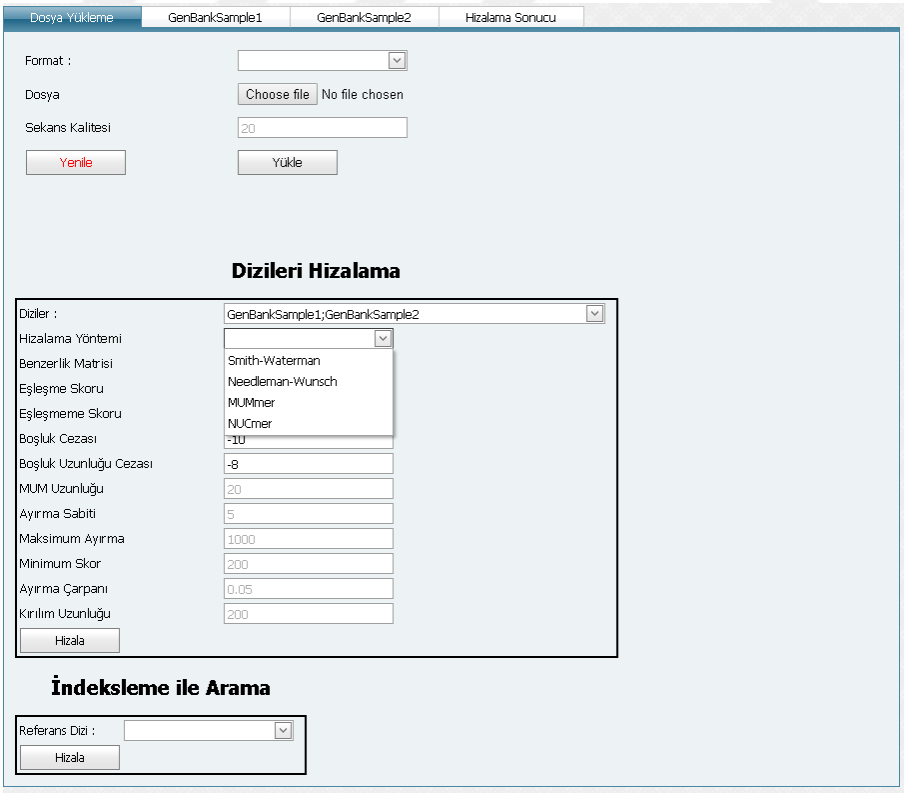
Şekil 5.7 Kullanıcı Girişi

Kullanıcı girişi yapıldıktan sonra kullanıcıların FATA, FASTAQ, GFF, GENBANK ve ABIF formatta dosyalarını yükleyebilecekleri ve yükledikleri bu dosyaların içeriğindeki dizileri hizalayabilecekleri Şekil 5.8'deki ekran açılmaktadır.

Dosya yükleme bölümünden diziler yüklendikten sonra "Tab" yazan bölümlere ve diziler listesine dosya isimleri gelmektedir. Listeden karşılaştırma yapılacak diziler seçilebilmektedir. Daha sonra hizalama yöntemleri bölümünden Şekil 5.9'daki gibi Smith Waterman, Needleman-Wunsch, MUMmer ve NUCmer yöntemlerinden biri seçilmektedir. Burada analizin amacına yönelik olarak hizalama yöntemi seçilmelidir. Örneğin büyük ölçekli dizilerin hizalanmasında alt yapısında indeksleme yöntemleri bulunan ve hızlı sonuçlar üreten MUMmer ve NUCmer yöntemleri seçilebilir (Delcher ve ark., 2002). Ayrıca Smith Waterman ve Needleman-Wunsch dinamik algoritmaları ile de kullanıcı lokal ve global hizalamalar yapabilmektedir.



Şekil 5.8 Anasayfa ekranı

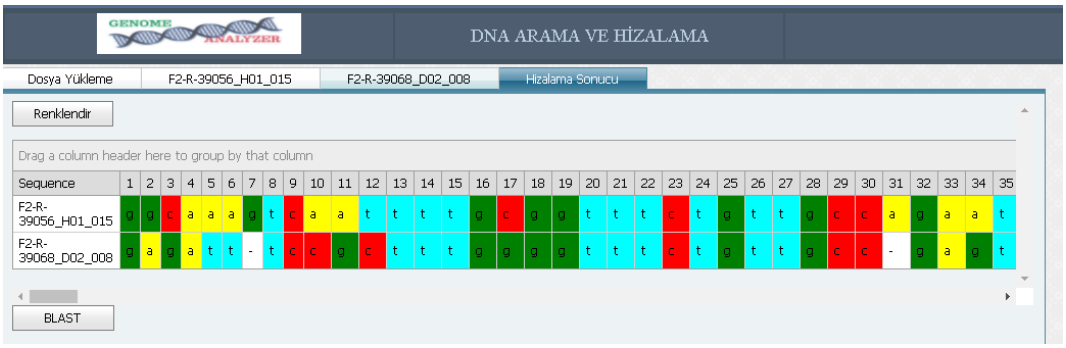


Şekil 5.9 Hizalama yöntemleri seçimi

Hizalama yöntemi seçildikten sonra benzerlik matrisi menüsünden AmbiguousDna, AmbiguousRna, Blosum45, Blosum50, Blosum62, Blosum80, Blosum90, Pam250, Pam30 ve Pam70 matrislerinden biri seçilmektedir. Genomik verilerin analizinde benzerlik matrisi seçiminin hizalama sonuçları üzerinde önemli etkisi vardır (Pearson, 2013). Hizalama algoritması ve dizilerin yapısına uygun olan benzerlik matrisi seçildikten sonra Eşleşme Skoru, Eşleşmeme Skoru, Boşluk Cezası, Boşluk Uzunluğu Cezası, MUM Uzunluğu, Ayırma Sabiti, Maksimum Ayırma, Minimum Skor, Ayırma Çarpanı ve Kırılım Uzunluğu gibi parametreler aktif olmaktadır. Bu parametreler hizalama skoru üzerinde etkiye sahip olduğundan dolayı genomik verilerin doğru hizalanabilmesi için parametrelerin seçimi önemlidir (Frith ve ark., 2010). Parametrelerin anlamları aşağıda verilmiştir.

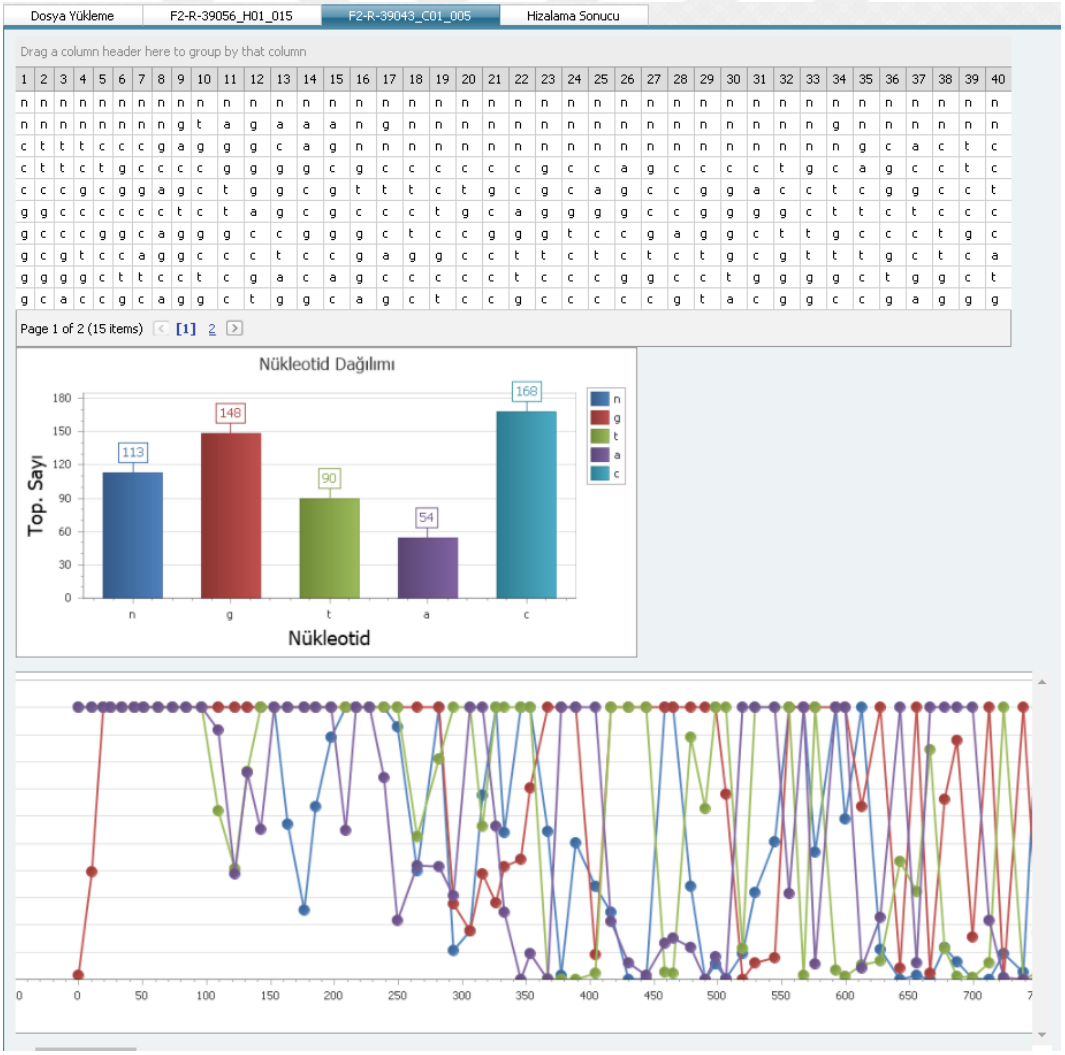
1. **Eşleşme Skoru:** Dizi elemanlarının eşleşme durumlarına verilecek skordur.
2. **Eşleşmeme Skoru:** Dizi elemanlarının eşleşmeme durumlarına verilecek ceza skorudur.
3. **Boşluk Cezası:** Dizilerde hizalama esnasında oluşacak boşluklara verilecek ceza skorudur.
4. **Boşluk Uzunluğu Cezası:** Boşluk uzunluğunun artması durumunda verilecek ceza skorudur.
5. **MUM Uzunluğu:** Sadece MUMmer ve NUCmer hizalama algoritmaları seçildiğinde aktif olan alt dizilerde hizalamanın olabilmesi için en az sahip olması gereken tam eşleme uzunluğu değeridir.
6. **Ayırma Sabiti:** Sadece NUCmer hizalama algoritması seçildiğinde aktif olan alt dizilerde hizalamanın olabilmesi için hizalamada oluşabilecek kırılma değerinin maksimum değeridir.
7. **Maksimum Ayırma: S**adece NUCmer hizalama algoritması seçildiğinde aktif olan kümelerdeki bitişik eşleşmeler arasındaki maksimum mesafe değeridir.
8. **Minimum Skor: S**adece NUCmer hizalama algoritması seçildiğinde aktif olan hizalamanın olabilmesi için minimum oluşabilecek skor değeridir.
9. **Ayırma Çarpanı:** NUCmer hizalama algoritması için ayırma çarpan değeridir.
10. **Kırılım Uzunluğu:** NUCmer hizalama algoritması için hizalamayı durdurmadan önce uzatılacak baz sayısını açıklar.

Anasayfada gerekli parametreler ayarlandıktan sonra hizala butonuna basıldığında Hizalama Sonucu sayfasında Şekil 5.10'daki renklendirilmiş hizalama sonucu görülecektir.



Şekil 5.10 Hizalama sonucu

Dizi isimlerinin yazılı olduğu sayfalar tıklandığında dizilerin içeriği, istatistiksel bilgileri ve sekans piklerinin görülebildiği Şekil 5.11'deki ekran açılacaktır. Burada görülen "n" karakterleri sekans kalitesi düşük olan verileri temsil etmektedir.



Şekil 5.11 Dizi içerik bilgileri

6. SONUÇ

Gelişen teknoloji ve moleküler biyoloji alanındaki ilerleyişle birlikte, araştırmacıların elinde büyük boyutlarda, deneysel veri birikmiştir ve artarak birikmeye devam etmektedir. Prof. Dr. Afig Berdeli yönetimindeki Ege Üniversitesi Tıp Fakültesi Çocuk Sağlığı ve Hastalıkları Ana Bilim Dalı'nda da DNA analiz cihazlarından alınan büyük miktarda veriler ile DNA analizleri yapılmaktadır. Özellikle yeni nesil dizileme tekniği ile DNA analizleri günümüzde büyük hız kazanmakla birlikte veri boyutunda da artış sağlamıştır. Bunlardan en sık kullanılan teknolojiler llumina Miseq/Hiseq/GA baz senteziyle dizileme yöntemini ve Solid Sistemi (platformu) ligasyon yoluyla dizileme yöntemleridir.

Bu tez çalışmasında genomik veritabanları incelenmiş ve bu veritabanlarında etkili arama yapabilmek için günümüze kadar geliştirilen algoritmalar araştırılmıştır. DNA analizlerinde araştırmacılar için hız önemli bir faktör olduğundan bu algorimalardan özellikle indeksleme yöntemlerini kullananlar tercih edilmiştir. DNA analizlerinde arama ve indekslemenin yanında önemli bir yeri olan kümele algorimaları da araştırılmış ve yeni kümeleme yöntemleri önerilmiştir. Önerilen yöntemlerin sonuçları farklı çalışmalardan elde edilen sonuçlar ile karşılaştırılmıştır. Karşılaştırma sonuçlarına göre önerilen yöntemlerin literatürdeki diğer algoritmalardan daha iyi kümeleme yaptığı görülmüştür.

Ayrıca bu çalışmada Ege Üniversitesi Tıp Fakültesi Çocuk Sağlığı ve Hastalıkları Ana Bilim Dalı'nda kullanılan DNA analiz cihazının ürettiği verilerin işlenmesi ve analizi için web tabanlı bilgisayar yazılımı geliştirilmiştir.

KAYNAKLAR DİZİNİ

Watson, J. D., and Crick, F. H., 1953, Molecular structure of nucleic acids. Nature, 171(4356), pp. 737-738.

Azuaje, F. and Dopazo, J., 2005, Data Analysis and Visualization in Genomics and Proteomics, John Wiley & Sons.

Alkan, C., Kavak, P., Somel, M., Gokcumen, O., Ugurlu, S., Saygi, C., ve Özören, N., 2014, Whole genome sequencing of Turkish genomes reveals functional private alleles and impact of genetic interactions with Europe, Asia and Africa, BMC genomics, 15(1), 963.

Ammari, H., 2008, An Introduction to Mathematics of Emerging Biomedical Imaging, Springer.

Luscombe, N. M., Greenbaum, D., and Gerstein, M., "What is bioinformatics? An introduction and overview, 2001, Yearbook of Medical Informatics, 1(83-100), 2.

Reichhardt T., It's sink or swim as a tidal wave of data approaches, 1999, Nature, 399(6736), pp. 517-20

Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp, B.A., and Wheeler, D.L., 2000, GenBank, Nucleic Acids Res, 28 (1), pp.15-8.

Bairoch, A. and Apweiler, R., 2000, The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000, Nucleic Acids Res, 28(1), pp. 45-48.

GENBANK İstatistik, https://www.ncbi.nlm.nih.gov/genbank/statistics/ (Erişim Tarihi: 10 Temmuz, 2017).

Telefoncu, A., Küfrevioğlu, Ö.İ., ve Pazarlıoğlu, N., 2003, BİYOİNFORMATİK-I, Ege Üniv. Matbaası, İzmir, s. 237.

Luscombe, N., Greenbaum, D., and Gerstein, M., 2001, What is bioinformatics? An introduction and overview, New Haven, USA: Yearbook of Medical Informatics.

Genbank, http://www.ncbi.nlm.nih.gov/genbank (Erişim Tarihi: 10 Temmuz, 2017)

Kneale, G., and Kennard O., 1984, The EMBL nucleotide sequence data library, Biochem Soc Trans 12(6), pp.1011–1014

KAYNAKLAR DİZİNİ (DEVAM)

Tateno Y., Imanishi T., Miyazaki S., Fukami-Kobayashi K., Saitou N. and Sugawara H., et al, 2002, DNA data bank of Japan (DDBJ) for genome scale research in life science, Nucleic Acids Res 30(1), pp.27–30.

Lesk, A.M., 2002, Introduction to Bioinformatics, Oxford University Press.

Setubal, J., and Meidanis, J., 1997, Introduction to Computational Molecular Biology**,** PWS Publishing Company**.**

Bairoch A., and Apweiler R., 1999, The UniProt protein sequence data bank and its supplement TrEMBL in 1999, Nucl Acids Res, 27, pp.49–54.

Xu, D., 2012, Protein databases on the internet, Current protocols in protein science, pp. 2-6.

Suzek, B. E., Huang, H., McGarvey, P., Mazumder, R., and Wu, C. H., 2007, UniRef: comprehensive and non-redundant UniProt reference clusters, Bioinformatics, 23(10), pp. 1282-1288.

Leinonen, R., Diez, F. G., Binns, D., Fleischmann, W., Lopez, R., and Apweiler, R. ,2004, UniProt archive. Bioinformatics, 20(17), pp. 3236-3237.

McGarvey P. B., Huang H., Barker W. C., Orcutt B. C., Garavelli J. S., and Srinivasarao G.Y., et al, 2000, PIR: a new resource for bioinformatics, Bioinformatics, 16(3), pp.290-291.

Hunter, L., 1993, Artificial Intelligence and Molecular Biology, AAAI Press.

Chengalvala, M. V., Chennathukuzhi, V. M., Johnston, D. S., Stevis, P. E., and Kopf, G. S., 2007, Gene expression profiling and its practice in drug development, Current genomics, 8(4), pp. 262-270.

Barrett, T., and Edgar, R., 2006, Mining microarray data at NCBI's Gene Expression Omnibus (GEO), Gene Mapping, Discovery, and Expression: Methods and Protocols, pp. 175-190.

Swiss Institute of Bioinformatics, Uniprotkb/swiss-prot protein knowledgebase release 2017/07 statistics, http://web.expasy.org/docs/relnotes/relstat.html, (Erişim Tarihi: 25 Temmuz, 2017)

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J., 1990, Basic local alignment search tool, Journal of molecular biology, 215(3), pp. 403-410.

KAYNAKLAR DİZİNİ (DEVAM)

Levenshtein, V. I., 1966, Binary codes capable of correcting deletions, insertions, and reversals, In Soviet physics doklady, Vol. 10, No. 8, pp. 707-710.

Gollery, M., 2004, Bioinformatics: Sequence and Genome Analysis, David W. Mount. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press, 692 pp., Clinical Chemistry, 51(11), pp. 2219-2219.

Mullan, L., 2006, Pairwise sequence alignment—it's all about us!. Briefings in bioinformatics, pp. 113-115.

Needleman, S. B., and Wunsch, C. D., 1970, A general method applicable to the search for similarities in the amino acid sequence of two proteins, Journal of molecular biology, 48(3), pp. 443-453.

Smith, T. F., and Waterman, M. S., 1981, Identification of common molecular subsequences, Journal of molecular biology, 147(1), pp. 195-197.

Krane D.E., and Raymer M.L., 2002, Fundamental Concepts of Bioinformatics, Benjamin Cummings.

Notredame, C., 2007, Recent evolutions of multiple sequence alignment algorithms, PLoS computational biology, 3(8), e123.

Edgar, R. C., 2004, MUSCLE: multiple sequence alignment with high accuracy and high throughput, Nucleic acids research, 32(5), pp. 1792-1797.

Edgar, R. C., and Batzoglou, S., 2006, Multiple sequence alignment, Current opinion in structural biology, 16(3), pp. 368-373.

Salzberg, S. L., 1998, Decision trees and Markov chains for gene finding, Computational Methods in Molecular Biology, Amsterdam, pp. 187-203.

Manber, U., and Myers, G., 1993, Suffix arrays: a new method for on-line string searches, SIAM JComput, 25(5), pp. 935–948.

Gusfield, D., 1999**,** Algorithms On Strings, Trees, and Sequences: Computer Science and Computational biology, Cambridge University Press.

Manning, C.D., Raghavan, P., and Schütze, H., 2008**,** Introduction to Information Retrieval, Cambridge University Press.

Bayer, R., and McCreight, E., 2002, Organization and maintenance of large ordered indexes, In Software pioneers, pp. 245-262.

Weiner, P., 1973, Linear pattern matching algorithms, In: Proceedings of 14th IEEE symposium on switching and automata theory, pp. 1–11.

KAYNAKLAR DİZİNİ (DEVAM)

Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O., and Salzberg, S. L., 1999, Alignment of whole genomes, Nucleic acids research, 27(11), pp. 2369-2376.

MacQueen, J., 1967, Some methods for classification and analysis of multivariate observations, In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, No. 14, pp. 281-297.

Theodoridis, S., and Koutroumbas, K., 2006, Pattern recognition, 3rd. San Diego, CA : Academic Press.

Anderberg, M.R., 1973, Cluster Analysis for Applications, Academic Press Inc.

Xu, R., and Wunsch, D.C., Clustering, IEEE Press, New Jersey, 2009, pp. 358.

Bagirov, A.M., Modified global k-means algorithm for minimum sum-of-squares clustering problems, Pattern Recognition, 41(10), pp. 3192-3199.

Jain, A.K., and Dubes, R.C., 1988, Algorithms for Clustering Data, Prentice Hall, New Jersey, 320 p.

Tanır D., Sadık T., and Nuriyev U., 2017, A mathematıcal model for density-based clustering methods, Proceeding of the International scientific conference on "Theoretical and application problems of Mathematics", Azerbaijan, Sumgait State University, May 22-26, pp. 206-207.

Kim, S. Y., Lee, J. W., and Bae, **J. S.,** 2006, Effect of data normalization on fuzzy clustering of DNA microarray data, BMC Bioinformatics, 7(1), 134.

Khan, S. S., and Ahmad, A., 2004, Cluster center initialization algorithm for K-means clustering, Pattern recognition letters, 25(11), pp. 1293-1302.

Erisoglu M., Calis N., and Sakallioglu S., 2011, A new algorithm for initial cluster centers in k-means algorithm, Pattern Recognition Letters, 32(14), pp. 1701-1705.

UCI Repository, http://archive.ics.uci.edu/ml/datasets.html (Erişim Tarihi: 15 Ağustos, 2017)

Bandyopadhyay, S., Mukhopadhyay, A., and Maulik, U., 2007, An improved algorithm for clustering gene expression data, Bioinformatics, 23(21), pp. 2859-2865.

KAYNAKLAR DİZİNİ (DEVAM)

Maulik, U., and Bandyopadhyay, S., 2003, Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification, IEEE Trans. Geosci. Remote Sens., 41, pp.1075–1081.

Jain, A. K., and Dubes, R.C., 1988, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ.

Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., and Golub, T. R., 1999, Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation, Proceedings of the National Academy of Sciences, 96(6), pp. 2907-2912.

Tanır, D., and Nuriyeva, F., An effective method determining the initial cluster centers for K-means for clustering gene expression data, 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 2017, pp. 751-754, doi: 10.1109/UBMK.2017.8093520.

Tanır, D. and Nuriyeva, F., 2017, On selecting the Initial Cluster Centers in the K-means Algorithm, Application of Information and Communication Technologies (AICT2017), Moscow, Russia, pp. 131-135.

Tanır, D., Sadık, T., and Nuriyev, U., 2017, A mathematical model for density-based clustering methods, Proceeding of the International scientific conference on Theoretical and application problems of Mathematics, Azerbaijan, Sumgait State University, pp. 206-207.

Acosta, B. D., 2011, Microsoft Biology Initiative:. NET Bioinformatics Platform and Tools, Journal of biomolecular techniques: JBT, 22(Suppl), S29.

Microsoft Biology Foundation, https://github.com/dotnetbio/bio (Erişim Tarihi: 20 Ağustos, 2017)

Pearson, W. R., 2013, Selecting the right similarity scoring matrix, Current protocols in bioinformatics, pp. 3-5.

Delcher, A. L., Phillippy, A., Carlton, J., and Salzberg, S. L., 2002, Fast algorithms for large-scale genome alignment and comparison. Nucleic acids research, 30(11), pp. 2478-2483.

Frith, M. C., Hamada, M., and Horton, P., 2010, Parameters for accurate genome alignment. BMC bioinformatics, 11(1), 80.

ÖZGEÇMİŞ

1989 yılında Muğla'nın Fethiye ilçesinde doğdum. İlköğrenimimi babamın işi sebebiyle Ankara, Malatya ve son olarak İzmir'de okuyarak tamamladım. 2007 yılında İzmir (YDA) Hürriyet Lisesi'nden mezun olduktan sonra yüksek öğrenimimi Ege Üniversitesi Matematik Bölümü Bilgisayar Bilimleri Opsiyonunda 2011 yılında tamamladım. 25 Kasım – 28 Aralık 2008 tarihleri arasında İzmir Takev Fen ve Anadolu Lisesi Kampüsü'nde, Centre Science Tasarımı ve Unesco'nun insiyatifiyle sergilenen uluslar arası interaktif "Niçin Matematik?" Sergisi'nde görev aldım. Kasım – Aralık 2009 tarihleri arasında düzenlenen 40 saatlik netsis ön muhasebe seminerine katıldım. Haziran – Ekim 2012 tarihleri arasında Work and Travel ile Amerika'da Yurt Dışı Deneyimim oldu. 2012-2014 yılları arasında Bilge Adam MCPD Yazılım Uzmanlığı ve Sistem ve Ağ Uzmanlığı eğitimi aldım. 2011-2012 sezonunun güz döneminde Ege Üniversitesi Matematik Bölümü Bilgisayar Bilimleri alanında başladığım yüksek lisans eğitimimi "DNA DİZİ ANALİZ CİHAZINDAN KAYNAKLANAN BELİRSİZLİKLERİN YAPAY ZEKA TEKNİKLERİ KULLANILARAK ORTADAN KALDIRILMASI" isimli tez çalışmam ile 2013 Temmuz ayında tamamladım. Aynı yılın içerisinde Eylül ayında doktora eğitimime TUBİTAK burs desteği de alarak başladım. Halen doktora eğitimim devam etmektedir.

YAYINLAR

Makaleler

A) Uluslararası hakemli dergilerde yayınlanan makaleler (SCI & SSCI & Arts and Humanities)

1. Nuriyeva F., Güler A., **Tanır D.**, (2012),"Mathematical Computer Games Based on Modular Aritmetic", Proceeding of the fourth International Conference "Problems of Cybernetics and Informatics", PCI 2012, Section 2, "Intellectual Technology and Systems", Vol. 1, pp. 194-196, Baku, Azerbaijan, September 12-14, (IEEE Xplore Digital Library).

   http://pci2012.science.az/2/15.pdf

B) Ulusal hakemli dergilerde yayınlanan makaleler

1. **Tanır D.,** Ugurlu O., Kapar M., Nuriyev U., 2017, Birleşik Stok Kesme ve Patern Sıralama Problemi için Bir Sezgisel Algoritma, Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi, (kabul edildi).

Kongreler

A) Uluslararası bilimsel toplantılarda sunulan ve bildiri kitabında (*Proceedings*) basılan bildiriler

1. **Tanır D**., and Nuriyeva F., "An effective method determining the initial cluster centers for K-means for clustering gene expression data," 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, 2017, pp. 751-754. doi: 10.1109/UBMK.2017.8093520 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8093520&isnumber=8093363

2. Sadik T., **Tanir D**., and Nuriyeva F., "A new method for lossless compression of binary images," 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 2017, pp. 1-4. doi:10.1109/IDAP.2017.8090198 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8090198&isnumber=8090153

3. **Tanır D.,** Nuriyeva F., 2017, On selecting the Initial Cluster Centers in the K-means Algorithm, Application of Information and Communication Technologies (AICT2017) Moscow, Russia, 20-22 September 2017, pp. 131-135.

4. **Tanır D.**, Sadık T., Nuriyev U. "A mathematıcal model for density-based clustering methods", Proceeding of the International scientific conference on "Theoretical and application problems of Mathematics", Azerbaijan, Sumgait State University, May 22-26, pp. 206-207, 2017, (http://sdu.edu.az/userfiles/file/conferences/math_01022016.pdf).

5. Ugurlu O., **Tanir D.**, and Nuri E., "A better heuristic for the minimum connected dominating set in ad hoc networks," 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT), Baku, 2016, pp. 1-4. doi: 10.1109/ICAICT.2016.7991751 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7991751&isnumber=7991641

6. **Tanır D**., 2016, An Allignment Algorithm for DNA Sequences Using Quality Information, International Artificial Intelligence and Data Processing Symposium'16, Malatya, Türkiye, 17-18 Eylül 2016, pp. 485-488.

7. Ugurlu O., **Tanır D.,** 2016, A Hybrid Genetic Algorithm for Minimum Weight Dominating Set Problem, World Conference on Soft Computing Berkeley, May 22-25 2016, pp. 85-89.

8. **Tanir D.**, Nuriyev U., 2013, Verification of DNA Sequencing Errors Caused by DNA Analysis Device, The International IIE (Institute of Industrial Engineers) Conference, Istanbul, Turkey, June 26-28, pp. 159-159.

B). Ulusal bilimsel toplantılarda sunulan ve bildiri kitabında basılan bildiriler

1. **Tanır D.,** Ugurlu O., Nuriyev U. and Guler A. 2016, Çelik Endüstrisinde Kaşılaşılan İki Amaçlı Bir Boyutlu Stok Kesme Problemi Üzerine, Yöneylem Araştırması ve Endüstri Mühendisliği 36. Ulusal Kongresi, İzmir, Türkiye, 13-15 Temmuz 2016.

BURSLAR VE ÖDÜLLER

TUBITAK, Yurt içi Doktora Burs Programı – 2228, 2013-2017

EKLER

Ek 1 Gen Analiz Web Programı C# Programlama Dili Kodları

Ek 1 Gen Analiz Web Programı C# Programlama Dili Kodları

```
using Bio;
using Bio.Algorithms.Alignment;
using Bio.Algorithms.Assembly;
using Bio.Algorithms.MUMmer;
using Bio.Extensions;
using Bio.IO.Bed;
using Bio.IO.FastA;
using Bio.IO.FastQ;
using Bio.IO.GenBank;
using Bio.IO.Gff;
using Microsoft.Office.Core;
using Microsoft.Office.Interop.Excel;
using Microsoft.Vbe.Interop;
using BiodexExcel.Visualizations.Common;
using Bio.SimilarityMatrices;
using Bio.Web.Blast;
using Bio.Algorithms.SuffixTree;
namespace GenomAnalyzer
{
   public partial class Default : System.Web.UI.Page    {
       int s1;
      decimal s2;
  protected void Page_Load(object sender, EventArgs e)
      {
         Session["COMBO"] = "";
         cmbSeq.Items.Clear();
         foreach (ISequenceParser parser in SequenceParsers.All)     {
cmbSeq.Items.Add(parser.Name.ToUpper(System.Globalization.CultureInfo.Curr
entCulture), parser);           }
         cmbSeq.Items.Add("ABIF FORMAT");
         AddAlignersDropDown();
        if (!IsPostBack)
         {
            ASPxPageControl1.ActiveTabIndex = 0;           }
```

```
ASPxPageControl1.TabPages[1].Text = (string)Session["Tab1"];
ASPxPageControl1.TabPages[2].Text = (string)Session["Tab2"];
          try           {

         ASPxListBox lstbx = (ASPxListBox)cmbSeqList.FindControl("listBox1");
          lstbx.Items.Clear();
         lstbx.Items.Add("Tümü");
         lstbx.Items.Add((string)Session["Tab1"]);
         lstbx.Items.Add((string)Session["Tab2"]);
         lstbx.DataBind();
         cmbSeqRef.Items.Add((string)Session["Tab1"]);
         cmbSeqRef.Items.Add((string)Session["Tab2"]);
           }
           catch (Exception)
           {           }
       }
       private void AddAlignersDropDown()
       {
           cmbAligner.Items.Clear();
           foreach (ISequenceAligner aligner in SequenceAligners.All)
           {
               cmbAligner.Items.Add(string.Format(aligner.Name), aligner);
           }
       }
       public string FileUpload(string gfilename){
           string filefolder = Server.MapPath("ProductUpload/");
           string filename = System.IO.Path.GetFileName(gfilename);
           string filepath = filefolder + filename;
           Session["FilePath"] = filepath;
      try
           {
               if (!System.IO.Directory.Exists(filefolder))
               {
                   System.IO.Directory.CreateDirectory(filefolder);
               }
FileSeqUpload.PostedFile.SaveAs(filepath);
           }
```

```
    catch (Exception exc)
    {
        Response.Write("Hata : " + exc.Message);
    }
    return filepath;        }

private void YUKLE(string file)        {
var parser = SequenceParsers.All[cmbSeq.SelectedItem.Text == "ABIF
FORMAT" ? 0 : cmbSeq.SelectedIndex] as IParser;
    if (parser == null)
    {
        return;
    }
    int currentRow = 1;
    try
    {
        var sequenceParser = parser as ISequenceParser;
        ISequence sequence;
        if (sequenceParser != null)
        {
            string filename = FileUpload(FileSeqUpload.PostedFile.FileName);
            if (cmbSeq.SelectedItem.Text == "ABIF FORMAT")
            {
                Parser pars = new Parser(txtKalite.Number);
             sequence = pars.CreateConfSeq(filename).ToList()[0] as  ISequence;
            }
            else
            {
                sequenceParser.Open(filename);
                sequence = sequenceParser.Parse().ToList()[0];
            }
            this.ReadSequences(sequence, filename, ref currentRow);
            sequenceParser.Close();
        }
        else
        {
            var rangeParser = parser as ISequenceRangeParser;   }    }
```

```
    catch (Exception ex){           }       }
    System.Data.DataTable dtgrid = new System.Data.DataTable();
    System.Data.DataTable dtgrid_chart = new System.Data.DataTable();
    private void ReadSequences(ISequence sequence, string fileName, ref int
currentRow)
    {
      List<ISequence> liste = new List<ISequence>();
      if (Session["Liste"] != null)
      {
        liste = Session["Liste"] as List<ISequence>;
      }
      int sequenceCount = 0;
      sequence.ID = Path.GetFileNameWithoutExtension(fileName);
      bool isExist = false;
      foreach (TabPage item in ASPxPageControl1.TabPages)
      {
        if (item.Text == sequence.ID)
        {
          isExist = true;
        }
      }
      if (!isExist)
      {
        liste.Add(sequence);
      }
      sequenceCount++;
      for (int i = 1; i <= 40; i++)
      {
        dtgrid.Columns.Add(i.ToString(), typeof(string));
      }
      int cCount = -1;
      DataRow newRow = dtgrid.NewRow();
      foreach (char item in sequence.ToList())
      {
        cCount++;
        newRow[cCount] = item.ToString().ToLower();
```

```
if (cCount == 39)
        {
            dtgrid.Rows.Add(newRow);
            newRow = dtgrid.NewRow();
            cCount = -1;
        }
    }
    dtgrid.Rows.Add(newRow);
    dtgrid_chart.Columns.Add("VALUE", typeof(string));
    foreach (char item in sequence.ToList())
    {
        newRow = dtgrid_chart.NewRow();
        newRow[0] = item.ToString().ToLower();
        dtgrid_chart.Rows.Add(newRow);
    }
    if (Session["Table1"] != null)
    {
        Session.Add("Table2", dtgrid);
        Session.Add("Tab2", sequence.ID);
        Session.Add("Table2_CHART", dtgrid_chart);
        Session.Add("Table2_CHART2", sequence);
    }
    else          {
        Session.Add("Table1", dtgrid);
        Session.Add("Tab1", sequence.ID);
        Session.Add("Table1_CHART", dtgrid_chart);
        Session.Add("Table1_CHART2", sequence);
    }
    Session.Add("Liste", liste);
}
protected void btnYukle_Click1(object sender, EventArgs e)
{
  YUKLE(FileSeqUpload.PostedFile.FileName);
  Response.Redirect(HttpContext.Current.Request.Url.AbsoluteUri);
}
protected void btnbtnAlign_Click(object sender, EventArgs e)
{
    DoAlignment();        }
```

```
        private void DoAlignment(params object[] args)
        {
            List<ISequence> selectedSequences = new List<ISequence>();
            selectedSequences = Session["Liste"] as List<ISequence>;
            AlignerInputEventArgs alignerInput = new AlignerInputEventArgs();
if (alignerInput != null)
            {
                alignerInput.Sequences = selectedSequences;
             alignerInput.Aligner = SequenceAligners.All[cmbAligner.SelectedIndex]
as ISequenceAligner;
                OnRunAlignerAlgorithm(alignerInput);
      }
   }

 private void AssignAlignerParameter(ISequenceAligner sequenceAligner,
AlignerInputEventArgs alignerInput)
        {
            alignerInput.Aligner.GapOpenCost =
Convert.ToInt32(txtGapCost.Number);
            alignerInput.Aligner.GapExtensionCost =
Convert.ToInt32(txtGapExtCost.Number);
            if (cmbMatrix.SelectedIndex > 0)        {
                string similarityMatrixOption = cmbMatrix.SelectedItem.Text;
                if (Enum.IsDefined(

typeof(SimilarityMatrix.StandardSimilarityMatrix),similarityMatrixOption))
{
  SimilarityMatrix.StandardSimilarityMatrix matrix =
(SimilarityMatrix.StandardSimilarityMatrix)Enum.Parse(typeof(SimilarityMatrix.
StandardSimilarityMatrix), similarityMatrixOption,  true);
alignerInput.SimilarityMatrix = new SimilarityMatrix(matrix);
                }
            }
            else
            {
  alignerInput.SimilarityMatrix = new
DiagonalSimilarityMatrix(Convert.ToInt32(txtEslesme.Number),
Convert.ToInt32(txtEslesmeme.Number)); }
```

```
        if (sequenceAligner is NucmerPairwiseAligner)
        {
            var nucmer = sequenceAligner as NucmerPairwiseAligner;
            nucmer.LengthOfMUM = Convert.ToInt32(txtLenMum.Number);
           nucmer.FixedSeparation = Convert.ToInt32(txtFixSep.Number);
            nucmer.MaximumSeparation = Convert.ToInt32(txtMaxSep.Number);
            nucmer.MinimumScore = Convert.ToInt32(txtMinScore.Number);
            nucmer.SeparationFactor = Convert.ToInt32(txtSepFact.Number);
            nucmer.BreakLength = Convert.ToInt32(txtBreakLen.Number);
        }
        else if (sequenceAligner is MUMmerAligner)
        {
            var mummer = sequenceAligner as MUMmerAligner;
            mummer.LengthOfMUM = alignerInput.LengthOfMUM;
        }
    }
    private void OnRunAlignerAlgorithm(AlignerInputEventArgs alignerInput)
     {
       AssignAlignerParameter(alignerInput.Aligner, alignerInput);
        try
        {
            IList<ISequenceAlignment> alignedResult =
alignerInput.Aligner.Align(alignerInput.Sequences);
            foreach (ISequenceAlignment sequence in alignedResult)
            {
                foreach (IAlignedSequence alignedSequence in
sequence.AlignedSequences)
                {
                    dtgrid.Columns.Add("Sequence", typeof(string));

                    for (int i = 1; i < alignedSequence.Sequences.Max(s => s.Count) +
1; i++)         {
                   dtgrid.Columns.Add(i.ToString(), typeof(string));
                    }
                    foreach (ISequence currSeq in alignedSequence.Sequences)
                    {
                      int cCount = -1;
```

```
                    DataRow newRow = dtgrid.NewRow();
                    foreach (char item in currSeq.ToList())

                    {     cCount++;
                       if (cCount == 0)
                       {
                          newRow[cCount] = currSeq.ID;

                          cCount++;
}
                       newRow[cCount] = item.ToString().ToLower();
}
                    dtgrid.Rows.Add(newRow);
                  }
                }
                Session.Add("TableAlign", dtgrid);              }
            Response.Redirect(HttpContext.Current.Request.Url.AbsoluteUri);
          }

          catch (Exception ex)
          {
             clsModule.ShowAlert(this, ex.Message);
          }
       }
 protected void cmbAligner_SelectedIndexChanged(object sender, EventArgs e) {
 LoadAlignmentArguments(cmbAligner.SelectedItem.Text);
ISequenceAligner sequenceAligner =
SequenceAligners.All[cmbAligner.SelectedIndex] as ISequenceAligner;
          if (sequenceAligner is NucmerPairwiseAligner)          {

             txtLenMum.Enabled = true;
             txtFixSep.Enabled = true;
             txtMaxSep.Enabled = true;
             txtMinScore.Enabled = true;
             txtSepFact.Enabled = true;
             txtBreakLen.Enabled = true;
          }
          else if (sequenceAligner is MUMmerAligner)
```

```
        {
            txtLenMum.Enabled = true;
        }
        else
        {
            txtLenMum.Enabled = false;
            txtFixSep.Enabled = false;
            txtMaxSep.Enabled = false;
            txtMinScore.Enabled = false;
            txtSepFact.Enabled = false;
            txtBreakLen.Enabled = false;
            txtLenMum.Enabled = false;
        }
    }
    private void LoadAlignmentArguments(string algoName)
    {
        IAlignmentAttributes alignmentAttributes =
this.GetAlignmentAttribute(algoName);
        if (null != alignmentAttributes)
        {
            foreach (KeyValuePair<string, AlignmentInfo> attribute in
alignmentAttributes.Attributes)
            {
                if (attribute.Value.DataType.Equals(AlignmentInfo.StringListType))
                {
                  this.CreateComboField(  attribute.Value, attribute.Key);
                }
}
  }
    }
    private string GetDefaultSM(IAlphabet sequenceAlphabet)
    {
     return sequenceAlphabet == Alphabets.DNA ?
    SimilarityMatrix.StandardSimilarityMatrix.AmbiguousDna.ToString()
    : (sequenceAlphabet == Alphabets.RNA     ?
   SimilarityMatrix.StandardSimilarityMatrix.AmbiguousRna.ToString()
    : (sequenceAlphabet == Alphabets.Protein  ?
SimilarityMatrix.StandardSimilarityMatrix.Blosum50.ToString()
```

```
SimilarityMatrix.StandardSimilarityMatrix.AmbiguousDna.ToString()));        }
private void CreateComboField(AlignmentInfo alignmentAttribute, string tag)
        {
          StringListValidator validator = alignmentAttribute.Validator as
StringListValidator;
           cmbMatrix.DataSource = validator.ValidValues;
           cmbMatrix.DataBind();
           cmbMatrix.SelectedIndex =
validator.ValidValues.ToList().IndexOf(alignmentAttribute.DefaultValue);
           if (alignmentAttribute.Name == "Similarity Matrix")
           {
               List<ISequence> liste = new List<ISequence>();
               if (Session["Liste"] != null)
               {
                   liste = Session["Liste"] as List<ISequence>;
               }
               cmbMatrix.SelectedIndex =
validator.ValidValues.ToList().IndexOf(GetDefaultSM(liste.ToList()[0].Alphabet
));
}
        }
       private IAlignmentAttributes GetAlignmentAttribute(string algoName)
       {
           IAlignmentAttributes alignmentAttributes = null;
           ISequenceAligner sequenceAligner =
SequenceAligners.All[cmbAligner.SelectedIndex] as ISequenceAligner;
           if (sequenceAligner is NucmerPairwiseAligner)
           {
               alignmentAttributes = new NUCmerAttributes();
           }
           else if (sequenceAligner is MUMmerAligner)
           {
               alignmentAttributes = new MUMmerAttributes();
           }
           else if ((sequenceAligner is SmithWatermanAligner) || (sequenceAligner is
NeedlemanWunschAligner)    || (sequenceAligner is PairwiseOverlapAligner))
```

```
        {
            alignmentAttributes = new PairwiseAlignmentAttributes();
        }
        return alignmentAttributes;
    }
    protected void cmbMatrix_SelectedIndexChanged(object sender, EventArgs
e)
    {
        if (cmbMatrix.SelectedIndex == 0)
        {
            txtEslesme.Enabled = true;
            txtEslesmeme.Enabled = true;
        }
        else
        {
           txtEslesme.Enabled = false;
            txtEslesmeme.Enabled = false;
        }
    }
    protected void btnSearch_Click(object sender, EventArgs e)
    {
        List<ISequence> liste = new List<ISequence>();
        if (Session["Liste"] != null)
        {
            liste = Session["Liste"] as List<ISequence>;
        }
        MultiWaySuffixTree MWsuffix =
        new MultiWaySuffixTree(liste[cmbSeqRef.SelectedIndex]);
        MWsuffix.MinLengthOfMatch = 40;
         List<Match> match= MWsuffix.SearchMatches(liste[1]).ToList();
    }
    protected void btnYenile_Click(object sender, EventArgs e)
    {
        Session["Liste"] = null;
        Session["TableAlign"] = null;
        Session["Table1"] = null;
        Session["Tab1"] = null;
```

```
            Session["Table1_CHART"] = null;
            Session["Table1_CHART2"] = null;
           Session["Table2"] = null;
           Session["Tab2"] = null;
           Session["Table2_CHART"] = null;
           Session["Table2_CHART2"] = null;
          Response.Redirect(HttpContext.Current.Request.Url.AbsoluteUri);
      }
    }
}
```