

Tartalomjegyzék

1.	Bevezetés.....	3
2.	Biológiai háttér	4
2.1	Sejtmag, kromoszóma	4
2.2	DNS	5
2.3	Gén, genom.....	5
2.4	A változás forrásai	6
2.5	SNP.....	6
2.6	Genetikai betegségek	7
2.6.1	Monogénés genetikai betegségek	7
2.6.2	Multifaktoriális és poligénés betegségek	7
2.6.3	Kromoszómális rendellenességek	8
3.	Bioinformatika.....	8
3.1	A bioinformatika területei	8
3.1.1	Szekvenciák kezelése	9
3.1.2	Molekulák térbeli szerkezetének vizsgálata.....	9
3.1.3	Kölcsönhatási hálózatok	10
3.2	Genom projektek	10
3.3	Szekvencia összerendezés	12
3.3.1	Szekvencia illesztés	13
3.3.2	Szekvencia hasonlóság.....	15
3.4	BLAST.....	17
3.4.1	Seeding.....	17
3.4.2	Extension.....	18
3.4.3	Evaluation	19
3.5	Genomikai adatbázisok, szolgáltatások	20
3.5.1	NCBI Entrez Programming Utilities.....	21
3.5.2	NCBI BLAST Web Service.....	21
3.5.3	BioMart	22
4.	Feladat-specifikáció	23
5.	A megvalósítás lehetséges módjai	24
5.1	Munkafolyamat.....	24
5.2	Magas szintű folyamatleírók, elosztott rendszerek.....	25
5.2.1	Tapasztalatok a Taverna rendszer használatával kapcsolatban	25

5.3	Saját rendszer fejlesztése	28
6.	Rendszerterv és rendszerműködés.....	28
6.1	Architektúra	29
6.2	Komponensek	30
6.2.1	Adatbázis absztrakciós réteg.....	30
6.2.2	HTTP kommunikáció.....	32
6.2.3	Naplózás.....	35
6.2.4	Felhasználó-azonosítás	35
6.2.5	NCBI Entrez kereső szolgáltatás.....	36
6.2.6	BLAST szolgáltatás	38
6.2.7	BioMart szolgáltatás	40
6.3	Adatstruktúra	41
6.4	Felhasználói felület.....	42
7.	A megvalósított rendszer	42
7.1	Implementálási környezet.....	43
7.2	Kutatási cél	43
7.3	A munkafolyamat bemutatása egy konkrét példa segítségével	45
7.3.1	Gén azonosítók lekérdezése	46
7.3.2	Azonosítók adatainak lekérdezése, feldolgozása	46
7.3.3	Ember → egér homológok lekérdezése	46
7.3.4	Találatok Entrez azonosítóinak meghatározása	47
7.3.5	Egér → ember homológok lekérdezése	47
7.3.6	Eredmények megjelenítése	48
7.4	Módosítások az eredeti elképzeléshez képest.....	48
7.5	Fejlesztési környezet.....	49
7.6	Futtatási környezet.....	49
7.7	Futási idők	50
7.8	Kutatási eredmények	50
8.	További perspektívák.....	53
9.	Összefoglalás	54
10.	Irodalomjegyzék	55
11.	Ábrajegyzék	57
12.	Abstract	58

1. Bevezetés

Néhány évtizeddel ezelőtt, 1953-ban jelent meg a Nature című folyóiratban a mára már legendássá vált cikk J. D. Watson és F. H. C. Crick [1] tollából, amely alapvetően a DNS szerkezetével foglalkozott, de a cikk végén egy nagyon elegáns utalás található arra vonatkozóan, hogy ez a nagyon bonyolult molekula a földi élet egyik örökítő anyaga lehet. Ma már tudjuk, hogy ez nem csak sejtés, a biológiai információ átörökítésében a nukleinsavaknak kulcsfontosságú szerepük van. Ugyanebben az évben, 1953-ban jelent meg az első hirdetés, amely tranzisztort tartalmazó berendezést (hallókészüléket) árult. Az akkor még meglehetősen nagyméretű tranzisztorok hamarosan a DNS kutatáshoz hasonlóan igen nagy érdeklődésre tettek szert. Egy emberöltőnyinél is kevesebb idő elteltével ma már a számítógépek világát éljük és majdnem teljes pontossággal ismerjük fajunk genetikai térképét.

A biológiának önálló szakterületeként fejlődött ki a molekuláris biológia, amely nagyon rövid idő alatt az érdeklődés és a kutatások középpontjába került. Az embereket igen erős szándék vezérli, hogy önmaguk működését minél jobban megértsék. Ezeknek a folyamatoknak a molekuláris szintű leírásával foglalkozik a molekuláris biológia. Az elmúlt évtizedekben számos mérföldkő fémjelzi ezen tudományág fejlődését. A közismertek közül néhányat említve: 1978-ban Smith és Nathans Nobel-díjat kapott a restrikciós enzimek felismeréséért és alkalmazásáért. 1980-ban Gilbert és Sanger részesült Nobel-díjban a DNS bázissorrendjének meghatározására kidolgozott módszeréért.

A számítógépek teljesítménye is nagy léptékkal fejlődött, a tárolási és feldolgozási kapacitás alkalmassá tette a számítógépeket biológiai problémák kezelésére is. Az 1980-as években a feldolgozott DNS-ek információit informatikai adatbázisokban tárolták, speciális biológiai vonatkozású algoritmusokat kezdtek el fejleszteni a bázissorrend becslésére, az egyes szekvenciák illesztésére. A fejlődés azóta is töretlen, manapság számos szabadon hozzáférhető adatbázis van az Interneten, amely nagyon sok kutatás eredményeit tartalmazza és ezek jól strukturáltan hozzá férhetőek.

Ennek a dolgozatnak a témája egy olyan informatikai rendszer tervezése és megvalósítása, amely biológiailag releváns adatok alapján próbál meg új információkat kinyerni már meglévő adatbázisokból. Ez azért lehetséges, mert mára annyi adat halmozódott fel, hogy az ezek között lévő kapcsolatokat nem láthatjuk pontosan, de az adatbányászat eszközeivel lehetőségünk nyílik arra, hogy ezeket az új információkat megtaláljuk.

A megvalósítandó feladat egy olyan adatbányászati szoftver létrehozása, amely külső biológiai adatbázisokkal kommunikálva egy adott folyamatot hajt végre. A lekérdezés során az adatokat elemzi, összehasonlítja és a folyamat szempontjából hasznosakat tárolja el. Ehhez rendelkezik lokális adatbázissal, amely a későbbi visszakereséseket is lehetővé teszi. A külső adatbázisokhoz való kapcsolódás körültekintő tervezése fontos, azért, hogy a felhasználás módja ne csak egy problémára megoldására legyen alkalmas. További megvalósítható feladat az is, hogy a folyamat ne a programba égetve jelenjen meg, hanem módosítható formában legyen tárolva, új folyamatokat lehessen definiálni. Összességében a cél ezen szempontok figyelembe vételével egy olyan széleskörűen használható bioinformatikai adatbányász szoftver létrehozása, amely nem csak egy adott probléma végrehajtását támogatja, hanem egy

olyan könnyen használható eszköz megvalósítása, amelyet kevés informatikai tudással, de nagy biológiai tudással rendelkező szakértők használhassanak.

Habár az elérni kívánt cél egy széles körűen alkalmazható rendszer kifejlesztése, a működés igazolásának szempontjából fontos, hogy egy konkrét példa segítségével végigvezessünk, bemutassunk egy munkafolyamatot. Absztrakt módon megfogalmazva: a rendelkezésünkre álló bemeneti adatokon végzett transzformáció segítségével új információt kapjunk a folyamat végén. Jelen megvalósításban genetikailag örökletes betegségekkel foglalkozunk, és arra vagyunk kíváncsiak, hogy van-e olyan hozzáférhető adat, amelyet még nem hoztak összefüggésbe emberekkel kapcsolatos kutatásokkal. Ha léteznek ilyen ismeretek, akkor azok bizonyos megfontolások szerint lehetséges, hogy a jövőben új kutatások alapját képezhetik. Ezek az emberek számára hasznosak lehetnek a vizsgált betegségek megértésével, és ez által kezelésével, gyógyításával kapcsolatban is.

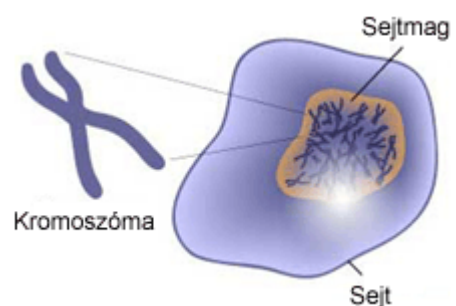
2. Biológiai háttér

Ahhoz, hogy a továbbiakban egyértelmű legyen a rendszer leírása, néhány biológiai vonatkozású fogalmat meg kell határozni. A megvalósítandó folyamatban az egyes lépések egymáshoz való viszonyát, a rendszer működésének célját csak úgy érthetjük meg, ha ezekkel a fogalmakkal tisztában vagyunk. Az itt leírt fogalmak a biológia által jelenleg elfogadott meghatározásokat tartalmazzák, azonban természetesen a jövőben az egyes fogalmak a felfedezéseknek köszönhetően új definíciót kaphatnak.

A vizsgálható kutatások olyan adatbázisokból származhatnak, amelyekben a releváns információ molekuláris szinten meghatározott és az öröklődéssel kapcsolatos, ezért az itt tárgyalt fogalmak többnyire a molekuláris- és sejtbológia témaköréből származnak. A fogalmak sorrendje azt a szemléletet követi, mintha egy mikroszkóp segítségével egyre nagyobb nagyítással szemlélnénk a sejt belsejét. Ezek után pedig a sejt számunkra érdekes működési mechanizmusai közül néhány és a vizsgált munkafolyamathoz kapcsolódó fogalmak kerülnek bemutatásra.

2.1 Sejtmag, kromoszóma

A sejtmag (1. ábra) vagy nukleusz az eukarióta sejtekben található, legtöbbször gömb alakú, néhány mikrométer átmérőjű képződmény. Ebben a membránnal elkülönített térben találhatóak a sejt kromoszómái, amelyek a sejtmag anyagának legnagyobb részét alkotják. [2] A kromoszómákat elkülönülten csak az osztódás során lehet észlelni, amikor azok feltekeredett állapotban vannak. Az osztódások közötti periódusban a sejtmag fehérje fonalrendszere fellazul, letekeredik és úgynevezett kromatinállományt alkot, amelyben az egyedi kromoszómák nem különülnek el. A kromatinállomány két fő alkotóeleme a DNS, és a hozzá szorosan kötődő hisztonfehérjék. A hisztonfehérjék a DNS feltekeredésében



1. ábra. A sejtmag a sejtben

játsszanak fontos szerepet, illetve képesek befolyásolni a génkifejeződést is (génexpresszió).

2.2 DNS

A DNS egy mozaikszó, a dezoxiribonukleinsav rövidítése. A prokarióta és eukarióta szervezetek és egyes vírusok genetikai információját az egymást követő nemzedékek között örökíti át. A kódolt információ nem csak a fehérjék szerkezetét határozza meg, hanem szabályozza azok időbeli és mennyiségi szintézisüket, így közvetett módon a sejt valamennyi funkciója a DNS ellenőrzése alatt áll.

Szerkezete lehetővé teszi az információ majdnem tökéletes tárolását, az adat másolását, átadását. Az előzőekben említetteknek megfelelően szerkezete a sejt egyes fázisaiban változik. Rendkívül nagyméretű nukleinsav, nukleotidok egységekből álló polimer. A nukleotid egységet egy cukor molekula (dezoxi-ribóz), egy foszfát csoport és egy bázikus nitrogéntartalmú heterociklus alkotja. A DNS esetében ez a gyűrűs molekula adenin (A), timin (T), citozin (C) és guanin (G) lehet. (2. ábra) Az információt a bázisok sorrendje határozza meg. Három egymás utáni bázis, egy triplet határoz meg egy aminosavat, amely a fehérjék alapegysége. Továbbá a tripletet, azaz az aminosav-kódoló szótár egy szavát kodonnak nevezzük. Mivel négyelemű a halmaz, amelyet a DNS tartalmazhat, és három hely kódol egy aminosavat és a sorrend számít, kiszámítható, hogy 64 lehetséges változat van. Azonban az élőlényekben csak húsz féle aminosav fordulhat elő. A kódszótár tehát redundáns, illetve bizonyos tripletok speciális jelentéssel bírnak. [3]



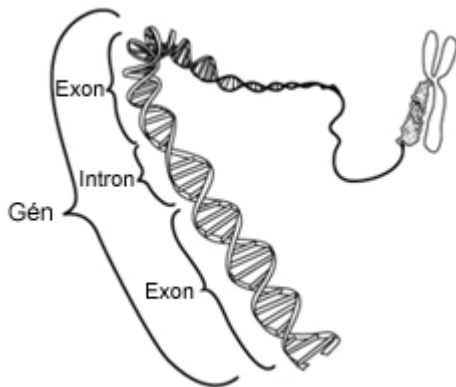
2. ábra. A DNS kettős spirál alakja és a bázispárok elhelyezkedése

A jól ismert kettős spirálban a két spirál egymásnak a „tükrképe”, ugyanazon a helyen lévő adeninnel szemben csak timin, citozinnal szemben csak guanin lehet, ezek a párokat bázispároknak nevezzük. Ezek a szabályok teszik lehetővé az információ nagyon pontos másolását, megőrzését.

2.3 Gén, genom

Szűkebb értelemben génnek nevezünk minden olyan DNS szakaszt (bázisszekvenciát), amely átíródhat ribonukleinsavakba, RNS-be. [4] Ezt a folyamatot transzkripciónak nevezzük. Az átíródott nukleinsavak a fehérjésintézist irányítják. Tágabb értelemben a gén részét képezik azok a szekvenciák is, amelyek nem íródnak át, de szabályozzák az átíródó szakasz működését. Egy gén számos tripletból áll, annyiból ahány aminosavat tartalmazni fog a gén által kódolt fehérje, illetve még néhányból, amelyek a szabályozásban vesznek részt. Genomnak a gének összességét nevezzük, amely értelemszerűen megegyezik a DNS mennyiségével. A humán genom összességében körülbelül $3 \cdot 10^9$ bázispárból áll, egy gént akár több tízezer kodon is

alkothat. Az aminosavakat kódoló géneken kívül nagy százalékban nem-kódoló



3. ábra. A gén a DNS egy szakasza, amelyet intron és exon részek alkotnak

szekvenciák találhatók. Ezt repetitív DNS-nek nevezik, mert sokszor ismétlődő szekvenciákat tartalmaz. A DNS-nek ez igen nagy részét teszi ki, régebben ezt hulladéknak tartották, mert közvetlen szerepe egyelőre nem ismert. Azonban az újabb kutatások egyik fontos területe ezen szekvenciák szerepének feltárása. A gének és ezáltal a DNS különböző részekre, úgynevezett intron és exon részekre osztható. Az intron egy gén olyan DNS szakasza, amely a gén által termelt fehérje egyetlen egy aminosavát sem kódolja. A kivágás után maradó régiókat exonoknak (3. ábra) nevezzük.

Fontos még megemlítenünk a génkifejeződés, azaz génexpresszió [5] fogalmát. A gének által kódolt fehérjék szintézisének mennyiségét az egyes sejtciklusokban a DNS más régiói szabályozzák. Ezáltal bár az összetett szervezetek DNS állománya minden sejtben megegyezik, de a sejtek differenciálódásban fontos szerepe van a génexpresszióknak.

2.4 A változás forrásai

Az előbbieken említett, hogy a DNS a majdnem tökéletesen alkalmas a tárolt szekvencia átörökítésére. A replikáció során a sejtnek vannak bizonyos mechanizmusai, amelyek elősegítik a DNS variálódását, változását. A másolás során előfordulhat hiba is, amikor egy bázispár rosszul íródik át. Valamint az osztódás során a kromoszómák egyes homológ régiókban kicserélődhetnek, ezt rekombinációnak nevezzük.

Fontos szerepe van a mutációnak, amely során az örökítő anyag spontán, külső körülmények vagy a véletlen hatására változik meg. Ezáltal egy új genetikai tulajdonság jön létre, amely lehet előnyös, illetve hátrányos is. A mutáció lehet pontszerű illetve érintheti az egész kromoszómát is. A külső körülmények erős megváltozása (hőmérséklet), mutagén anyagokkal való reakció (kemikáliák, gyógyszerek), környezeti hatások (radioaktív sugárzás, nehézfémzennyezés) nagymértékben járulhatnak hozzá a mutáció létrejöttéhez.

2.5 SNP

A kifejezés a Single Nucleotid Polymorphism, magyarul az Egyszerű Nukleotid Polimorfizmus rövidítése, „szip”-nek is szokás nevezni. Egy olyan DNS szekvencia variáció, amikor az egyedek között vagy egyeden belül a kromoszómapárok között az eltérés egyetlen bázispárban (adenin – timin, citozin – guanin) van. [6] Tehát pontmutációnak tekinthető, amelyek sikeresek (adaptív mutáció), így végül fontos részévé válnak a populációnak. Ez a pontszerű eltérés különböző allélek

megjelenéséhez vezethet. Allélnak a gén egy változatát nevezzük, a klasszikus genetika óta ismert fogalom. Például egy allél lehet felelős a szőrzet színéért.

A humán genetikai variációk 90%-ért a snipek felelősek, minden 100-300 bázispáronként megjelenik a humán genomban. Ezek a variációk egyaránt lehetnek a gének közötti nem kódoló részekben, illetve lehetnek a gének különböző területein is, így a befolyásuk lehet a szintézisre és a szintetizált fehérjék szerkezetére egyaránt. Jelentőségük nagy, mert ezek a változások hatással vannak arra, hogy az emberi szervezet hogyan reagál betegségekre, baktériumokra, vírusokra, kemikáliákra.

2.6 Genetikai betegségek

A genetikai betegséget, azaz a genetikai rendellenességet egy vagy több gén abnormális kifejeződése okozza. Ennek okai a fentebb tárgyalt mutáció, kromoszóma rendellenesség, kromoszóma szám változás, triplet sokszorozódás lehetnek. Amennyiben a gént az előző generációtól öröklí az egyed, abban az esetben beszélünk örökletes genetikai betegségről. A betegség különböző formákban jelenhet meg. Az öröklődés típusától függően lehetséges, hogy az egyed csak hordozó, de a betegség lehet domináns jellegű is, kifejeződése akár nemhez is köthető (például vérzékenység esetén). Jelenleg több ezer genetikai betegséget ismerünk. A betegségeket különböző csoportokba oszthatjuk, megkülönböztetünk egygénés, poligénés betegségeket illetve kromoszóma rendellenességeket. [7]

2.6.1 Monogénés genetikai betegségek

Az egygénés vagy monogénés betegségeket egy gén hibás működése okozza. Ez azt jelenti, hogy a gén által szintetizált fehérje valamilyen feladatát nem megfelelően tudja ellátni. A változás lehet semleges (színvakság), de akár letális is. Egyes esetekben a betegség más betegségeket elleni védelmet okozhat. A betegség kötődhet nemi és testi kromoszómához egyaránt. Klasszikus példa a vérzékenység öröklődése, amely az emberben az X ivari kromoszómával öröklődik.

2.6.2 Multifaktoriális és poligénés betegségek

A genetikai betegségek lehetnek összetettek, multifaktoriálisak és poligénésnek, azaz több gén rossz működését sok esetben az életmód és a környezeti tényezők által kiváltott hatások együttesen határozzák meg. Ezek a fajta betegségek számos esetben családon belül öröklődnek, de a klasszikus Mendel-féle öröklődési típusokkal nem jellemezhetőek. A potenciális beteg betegségre való hajlamát ezért nehéz meghatározni. Továbbá ezen betegségeket nehéz tanulmányozni és kezelni, mert az egyes betegségek megjelenését kiváltó specifikus okok nem minden esetben egyértelműen azonosítottak. Számos betegség tartozik ide, többek között: asztma, cukorbetegség, epilepszia, hipertenzió (magas vérnyomás), mániákus depresszió, schizophrénia, szájpadhasadék (farkastorok), bizonyos szívrendellenességek, rák.

2.6.3 Kromoszómális rendellenességek

A kromoszómák számának vagy struktúrájának az egészséges egyedekhez képest történő megváltozását nevezzük kromoszómális rendellenességnek. Általában akkor a sejt osztódásakor léphet fel. Számbeli eltérés esetén például Down-kór jöhet létre. A struktúra megváltozása esetén a szerkezetben törlődések, duplikálódások, eltolódások és más jelenségek léphetnek fel. Ezek a kromoszóma rendellenességek általában egy ivarsejtet érintenek, az adott egyedre vannak kihatással, de nem öröklődnek.

3. Bioinformatika

A bioinformatikát többféle módon definiálják. A gyakrabban előforduló megközelítés szerint egy interdiszciplináris tudományág, amely a biológia, az informatika és az információ technológia határán helyezkedik el. [8] Más megközelítésben a bioinformatika a biológiai információk számítógépes kezelésének sajátos módját jelenti. Ez a fajta szemlélet azt vallja, hogy a tudomány minden ágában használnak informatikát, ezért ilyen értelemben határtudományról beszélni felesleges, ezáltal ez nem tekinthető önálló módszernek. Kijelenti továbbá, hogy mindegyik bioinformatikai irányzat által használt modellekben közös tulajdonság, hogy magukat a modelleket az adatok és a közöttük lévő relációk határozzák meg. [9]

Az merev interdiszciplinaritás helyett ez a fajta szemléletmód megfontolandó, hiszen egy tudományoknak rendelkezniük kell a saját paradigmáikkal, amelyek egyben meghatározzák a tudományos diszciplínák tényeinek, ismereteinek rendjét, valamint vezérlik a további lehetséges kutatási irányokat is. [10] A bioinformatika ilyen értelemben nem rendelkezik önálló paradigmával, azonban elmondható, hogy bizonyos biológiai problémák megoldásában az informatika fontos szerepet játszik. Informatikai szemszögből való megközelítés esetén az a tapasztalat, hogy a jelenlegi bioinformatikai alkalmazások nem feltétlenül igényelnek új absztrakciók befogadását, mint minden más gyakorlati megvalósítás esetén is, a problémák végül konkrét informatikai feladatokká válnak.

Fontos látni, hogy az 1980-as évektől kezdődően a mai napig igen intenzív fejlődés figyelhető meg a biotechnológia és a számítógép tudományok körében. A számítógépek teljesítményének fejlődése magával vonja a biotechnológiai problémák megoldhatóságának növekedést. A bioinformatikai szakterületnek az elmúlt időszakban önálló intézményei létesültek, folyóiratok jöttek létre, amelyek csak ezzel a témával foglalkoznak, az egyetemeken tárgyként vagy akár szakirányú képzésként is oktatják, évente több konferenciát rendeznek ebben a témában, valamint számos szak- és tankönyv látott a közelmúltban világot.

3.1 A bioinformatika területei

A vizsgált probléma szempontjából a bioinformatikai feladatok alapvetően három különböző irányzatra oszthatóak. Az egyik a DNS és a fehérjék szekvenciájával és azok adataival foglalkozik, a másik a molekulák térszerkezetével, a harmadik pedig a biológiai kölcsönhatások hálózataival foglalkozik.

3.1.1 Szekvenciák kezelése

A szekvenciákkal foglalkozó ágazat jelent meg először, a fejlődés sok tekintetben talán ezen a területen a leglátványosabb. Az egyes szekvenciák jól reprezentálhatók a számítógépen (karakterfüzerek), amelyek jól átláthatóak valamint a rajtuk végzett algoritmusok is fejlettek, köszönhetően annak, hogy ezekkel az adatstruktúrákkal való számolás viszonylag egyszerű. A szekvenciákkal kapcsolatos kutatások nagyon sok formát öltöttek, fontos megjegyeznünk, hogy korán megjelentek szekvencia adatbázisok. Az egyes elszigetelt adatbázisok az Internet megjelenésével könnyen átjárhatóvá váltak, és kialakultak olyan formátumok, amelyek az egyes szekvenciákhoz kapcsolódó annotációkat szabványosan kezelik, így a tudás jól kezelhetővé vált. Régebben, ha a kutató meg szeretne volna határozni egy gén vagy nukleotid szekvencia helyét a genomban, azt csak hosszú, fáradtságos munkával, manuálisan tudta megoldani. Manapság megjelentek olyan alternatívák, amelyek nagyméretű illesztéseket rövid idő alatt tesznek lehetővé. A feladat egy egyszerű, Internetes honlap kereső felületének használatává redukálódott, amivel nem csak időt lehet megtakarítani, de elképzelhető, hogy a keresést végző számára új, addig ismeretlen információ birtokába is jut, köszönhetően az adatbázisok fejlett keresőszolgáltatásainak. Az egyes adatbázisok fejlődésében fontos szerepet játszanak a genom projektek, amelyek többek között a szekvenált élőlények teljes genomjának információjával járulnak az adatbázisok bővüléséhez.

A szekvenciák kezelésével kapcsolatban fontos megemlíteni, a homológ szekvenciák vizsgálatát. A homológ ebben az értelemben azt jelenti, hogy a különböző fajok különböző génjei ugyanazt a célt szolgálják, de lehetséges, hogy a gének szekvenciájában eltérések lehetnek. Azaz a két gén egy közös őstől származik. Amennyiben két szekvencia ilyen értelemben megegyezik, akkor azokat homológoknak nevezzük. Ez a fajta felismerés számos következtetésre, kutatásra ad lehetőséget. Az egyes fajok között molekuláris szinten határozhatóak meg a rokonsági fokok. Az egyes eltérések „viselkedését” ismerve meghatározhatóvá válik, hogy időben a két gén mikor rendelkezett közös őssel. Természetesen ezek meghatározása igen nehéz, de a mutációk és a gén stabilitásának függvényében pontos becslések adhatóak. Így akár az ember és az élesztőgomba közötti hasonlóságok is kimutathatóak, illetve legkorábbi közös őssükkel kapcsolatos információk is meghatározhatóak. Ezekkel a fajta összefüggésekkel foglalkozó tudományágat, amely megpróbálja azonosítani és értelmezni a kapcsolatokat a földi élet különböző formái között, *filogenetikának* nevezzük.

3.1.2 Molekulák térbeli szerkezetének vizsgálata

A második irányzatot a molekulák térbeli szerkezetét megjósolni próbáló rendszerek alkotják. Bár a megközelítés nem teljesen biológiai alapú, ezek a fajta programok magukon hordozzák a fizika, a modellezési technikák egyes vonásait. A kiindulási alap az, hogy az egyes molekulák elsődleges szerkezetének ismeretében meg kell határozni a molekulák másodlagos, harmadlagos és negyedleges szerkezetét és ezt ábrázolni. A szerkezet meghatározása nehéz, mert jelenleg nem rendelkezünk a szükséges összes információval, amely alapján egy bármilyen tetszőleges struktúra térbeli modelljét ábrázolhatnánk. A jelenlegi kutatások heurisztikák segítségével próbálják meg áthidalni az ismeretek hiányát. Az ötlet azon az elgondoláson alapul,

hogy bizonyos egyszerűbb molekulák struktúráját ismerjük (hélixek, β -redők), valamint vannak ismereteink arról, hogy mik határozzák meg a magasabb szintű térbeli szerveződést illetve megpróbálunk kinyerni megfigyelésekből információkat. Ezeknek a



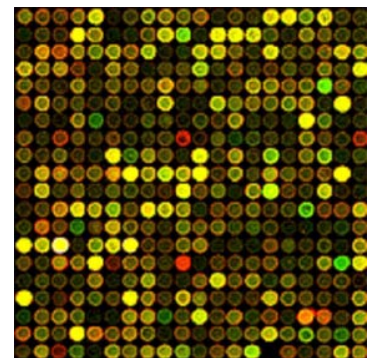
4. ábra. Számítógép által rajzolt molekula térszerkezet

próbálkozásoknak a jósága folyamatosan javul, a világon több versenyt is rendeznek, amelyeknek a célja bizonyos összetett molekulák minél élet hűbb ábrázolásához szükséges algoritmusok kifejlesztése. (4. ábra) Ennek az irányzatnak a jelentőségét többek között az adja, hogy amennyiben befolyásolni tudjuk a sejtben megjelenő fehérjék térbeli szerkezetét, akkor meg tudunk olyan fehérjét létrehozni, amelyek bizonyos általunk kitűzött feladatokat látnak el. Ez a gyógyszerektől kezdve a génmódosított növényeken keresztül sok mindent magába foglalhat. Kevésbé futurisztikus példát említve ez az irányzat jelenleg is fontos szerepet tölt be meglévő molekulákról röntgen kristallográfia, NMR spektroszkópia segítségével

készített felvételek elemzésében, értelmezésében.

3.1.3 Kölcsönhatási hálózatok

A legújabb kutatási irányzat a kölcsönhatási hálózatok vizsgálata. Megjelentek olyan kísérleti technikák (például microarray (5. ábra) és kettőshibrid módszerek), amely segítségével párhuzamosan akár több ezer gén expressziós szintje, nagyon sok ismert molekula (DNS, RNS, fehérje) együttes viselkedése nyomon követhetővé válik. Továbbá a folyamatosan fejlődő módszereknek köszönhetően számos fehérje és fehérjét nem kódoló RNS szabályozó szerepe megismerhetővé válik. Ezeknek a problémáknak kezelésére kézenfekvő megoldás a gráfok használata, amely témakörnek fejlett matematikai apparátusa áll rendelkezésre a felmerülő problémák kezelésére. Fontos lépés Erdős Pál és Rényi Alfréd véletlen gráf modellje [11], illetve fontos megemlíteni Barabási Albert-László munkásságát a skálafüggetlen hálózatokkal kapcsolatban [12], amelyek alkalmazása a mikro- és makroszintű biológiai hálózatok megértésében nagy szerepet töltenek be.



5. ábra. Microarray chip

3.2 Genom projektek

Az 1990-as évek elején határozták meg az első önálló életre képes élőlény teljes genomszerkezetét, nukleinsav-szekvenciáját. [13] Ez volt az első befejezett genom projekt. Visszatekintve láthatjuk, hogy ez az esemény nem egy egyedülálló kutatás végét jelentette, hanem egy új korszak kezdetét. Azóta számos élőlénynek határozták meg és tették elérhetővé a genomját. Számos baktérium, egysejtű eukarióta, növény, gerinctelen állat és az ember voltak az első alanyai a genom projektnek. Tehát a földi

élővilág legfontosabb nagy csoportjairól rövid időn belül genomikai ismeretek álltak rendelkezésre. Ezt biológiai oldalról az automatikus szekvenálási módszerek fejlődése, informatikai aspektusból a nagyméretű adatbázisok létrejötte támogatta. A mai napok trendjét a futószalagszerűen termelő genom szekvenálások jellemzik. Napjainkban már sok emlős, köztük emberszabású majmok, klasszikus kísérleti állatok, például egér, nyúl, muslica, patkány genomját ismerjük. A fejlődés töretlen, a jövőben az egyes archaikus leletek alapján lehetőség nyílna arra, hogy bepillantást nyerjünk kihalt állatok DNS szerkezetébe is.



6. ábra. A Nature 2001. február 15-én megjelent, 409. száma

A humán genom projekt (*Human Genome Project*) 1990-ben indult Amerikában az USA Department of Energy és National Institutes of Health irányításával. [14] A projekt célja meghatározni az emberi genomot, fizikai géntérképet készíteni róla és mindezt egy adatbázisban tárolni. A projekt megvalósítására szánt idő tizenhárom év volt. Az első hat évben géntérképek kialakítása történt. Majd 1996-tól az állami projekt keretén belül megkezdődött a szekvenálás. Az egyes hosszú nukleinsav láncok nem szekvenálhatóak, csak a viszonylag rövidebbek. Ezért, a kromoszómákat egyre kisebb szakaszokra osztják. Ezeket a darabokat vektorokba klónozza szekvenálják. 1998-tól megjelenik a verseny, az egyik kutató egy magáncéget alapít, és azt tűzi ki célul, hogy az államilag támogatott kutatásban használnál gyorsabb és olcsóbb módszert alkalmazzon. A cég által alkalmazott eljárás a *shotgun módszer*. A módszer lényege, hogy a nagyon hosszú, nehezen szekvenálható molekula darabokat véletlenszerűen kisebb, de egymást átfedő egységekre (*contigokra*) bontják szét és az így kapott kisebb szakaszokat már megbízhatóbban tudják kezelni. A folyamat végén az egyes darabokat számítógép segítségével illesztik össze. Bár már 2001-ben jelentős eredmények láttak napvilágot (6. ábra), a projekt 2003-ban fejeződött be. A mai napig jelennek meg újabb revíziók, legutóbb az első kromoszóma analízist fejezték be 2006 májusában. A folyamat még korántsem ért véget, habár az emberi DNS 99%-ának szekvenciáját ismerjük, a működést teljes mértékben nem értjük. A géneknek közelítően 60%-át ismerjük. Genom projektek eredményeként óriási tömegű szekvencia adat, megfeytetlen kód keletkezik, amelyek nyilvános és nem nyilvános adatbázisokba kerülnek. A bioinformatika a jövőben hasznos eszköz a szekvenciák által kódolt információ megfeytésében.

További konkrét példaként megemlíteném az egér genom projektet, (*Mouse Genome Project*) [15] amely a *Mus musculus* genom szekvenciájának feltérképezését tűzte ki célul, melyet a Mouse Genom Sequencing Consortium (MSC) koordinál. Hasonló projekt a Berkeley Drosophila Genome Project, mely az ecetmuslica *Drosophila melanogaster* genomját vizsgálja.

3.3 Szekvencia összerendezés

A szekvencia összerendezés egy olyan eljárási mód, amely során nukleinsavak, fehérjék elsődleges szerkezetét hasonlítjuk össze, feltételezve, hogy a hasonlóság működésbeli, strukturális vagy evolúciós kapcsolat következményeként áll fenn. [16] Az evolúciós változásoknak köszönhetően teljes mértékben izomorf szekvenciák előfordulásának valószínűsége igen csekély. Ez felveti azt a problémát, hogy két szekvenciát hogyan tudunk összehasonlítani. Bár az egyes szekvenciák informatikai reprezentációja, a szekvenciák karakterfüzérként való értelmezése, kézenfekvő megoldásokat kínál, ezek inkább sztring illesztési megoldások, amelyek nélkülözik a biológiai megfontolások figyelembevételét. Ebből következik, hogy az ilyen biológiai szekvenciák illesztését, a közöttük lévő kapcsolatok, hasonlóságok meghatározását csak biológiai eredetű ismeretek, heurisztikák segítségével adhatjuk meg.

Ahhoz, hogy a homológokat meg tudjuk határozni, figyelembe kell vennünk a nukleinsavak replikációjainak sajátosságait. A legegyszerűbb módosulás a pontmutáció, amely során egyetlen bázispár módosul egy másikra. Ezt elég könnyű észrevenni és kezelni, köszönhetően annak is, hogy elegendő ismerettel rendelkezünk a nagy gyakorisággal előforduló pontmutációkról. A nagyobb problémát a rések (*gaps*) jelentik. Ez azt jelenti, hogy az elsődleges szekvenciában olyan jelenségek fordulnak elő, amely során a szekvencia bizonyos része – a két faj közös őséhez vagy egymáshoz képest – eltűnik (törlődik), bizonyos ponton egy új szekvencia részlet ékelődik be. Ezeket a rész szekvenciákat nehéz meghatározni, amelyek után a szekvenciák közötti egyberendezés egyértelművé válik.

Alapvetően két fajta szemléletmódot különböztethetünk a szekvenciák összerendezésével kapcsolatban. Az egyik szerint a szekvenciákat illesztjük, amely során az illesztendő szekvenciát úgy transzformáljuk az illesztendő szekvenciához, hogy réseket helyezünk el az illesztendő szekvenciában, de az egyes alkotóelemeket nem módosítjuk. A másik fajta szemlélet figyelembe veszi azt a lehetőséget, hogy az egyes alkotóelemek az evolúció során átalakulhatnak más alkotóelemekké. Az egyes átalakulások, kicserélődések biológiai jelentőségét, relevanciáját figyelembe véve egy pontozási rendszert vezet be. Az illesztés akkor optimális, amikor a két szekvencia közötti hasonlóság kiszámítása után a lehető legtöbb pontszámmal rendelkezik. Ennek a fajta szemléletmódnak egyértelmű előnye mellett fontos megemlíteni hátrányát is, amely maga a módszer lényege. Minél pontosabb ismeretekkel kell rendelkezünk az egyes változásokról, hogy később azt egy algoritmikus formában megfelelően arányos skálár értékekké tudjuk alakítani.

Az egyes működési mechanizmusok mellett fontos szempont az összerendezési algoritmusok bonyolultsága is. Általánosságban elmondható, hogy polinomiális jellegű algoritmusokról van szó. A problémát alapvetően az adatmennyiség nagysága okozza. Az egyes szekvenciák illesztésekor az első nagyobb problémát annak a régiónak a megtalálása jelenti, ahol az adott szekvencia előfordul. Természetesen ez akár több régió is lehet, köszönhetően akár a paralóg szekvenciáknak. Szemléltetésképpen megemlíteném, hogy az emberi genom egyszerű szöveges fájlban tárolva mintegy két gigabájt lemezterületet igényel. A cél tehát az, hogy olyan algoritmust készítsünk, amely adott szekvenciák optimális illesztését, hasonlóságának vizsgálatát a viszonylag nagy adatmennyiség ellenére viszonylag kis idő alatt, de mégis biológiailag releváns módon végezze.

3.3.1 Szekvencia illesztés

A szekvencia illesztő eljárások, algoritmusok megjelenése nem sokkal a gének szekvenálásának elterjedése utáni időszakhoz köthető. Az első illesztéseket manuálisan hajtották végre, de ez hamar eltűnt, mert ez igen sok odafigyelést igénylő feladat, és a rendelkezésre álló szekvenált adatok mennyiségének növekedésével egyre lehetetlenebbé válik.

A legegyszerűbb megoldás a manuális összerendezés, amely során megpróbálunk ránézésre két szekvenciát illeszteni. Ez kis szekvenciák esetében kivitelezhető, az illesztett szekvenciákba réseket szúrunk be, figyeljük az egyezéseket és az eltéréseket. Egy olyan metrikát adunk meg, amely az előbb leírt szempontok figyelembevételével megadja a két szekvencia távolságát. A következő egyszerű példában az AACGTCCGA szekvenciához illesztjük az ACTCCA szekvenciát. (7. ábra)

A	A	C	G	T	C	C	G	A
A	-	C	-	T	C	C	-	A

7. ábra. Egyszerű manuális összerendezés.

Ezek után olyan technikák jelentek meg, amelyeket még mindig kézzel végeztek, azonban megpróbálták a hasonlóság felismerését könnyebbé tenni. Ilyen módszer a pontábrázolás (*dot-matrix*) [17], amely során a két szekvenciát egy mátrixban írták fel, úgy, hogy az egyik szekvencia volt a mátrix oszlopainak csúcsán, a másik szekvencia a mátrix sorainak bal szélén. Ezek után rendre összerendelték, hogy az egyes sorok által meghatározott alkotóelem melyik oszlop alkotóelemével egyezik meg. Amennyiben a grafikusán szemléljük az ábrázolást, akkor az összefüggő minél hosszabb vonalak minél nagyobb hasonlóságra utalhatnak. (8. ábra)

	M	T	F	R	D	L	L	S	V	S	F	E	G	P	R	P	D	S	S	A	G	G
M	X																					
T		X																				
F			X							X												
R				X										X								
D					X												X					
L						X	X															
L						X	X															
S								X		X								X	X			
V									X													
S								X		X								X	X			
F			X								X											
E												X										
G													X								X	X
P														X		X						
R				X											X							
P														X		X						
O																						
S								X		X								X	X			
S								X		X								X	X			
A																				X		
G													X								X	X
G													X								X	X

8. ábra. Pontábrázolás.

Természetesen ez hosszabb szekvenciák esetén is elképzelhető, ebben az esetben egy képi megjelenítés alkalmazásával már valódi vonalakat kaphatunk.

Komolyabb informatikai alkalmazást jelent azonban a dinamikus programozás segítségével megvalósított globális és lokális illesztés. A részletek tárgyalása előtt két egyszerű vizuális példával szemléltetném a két fajta illesztés közötti szemléletbeli különbséget.

FTFTALILLAVAV F--TAL-LLA-AV COELACANTH P-ELICAN-- <i>globális illesztés</i>	FTFTALILL-AVAV --FTAL-LLAAV-- COELACANTH -PELICAN-- <i>lokális illesztés</i>
---	--

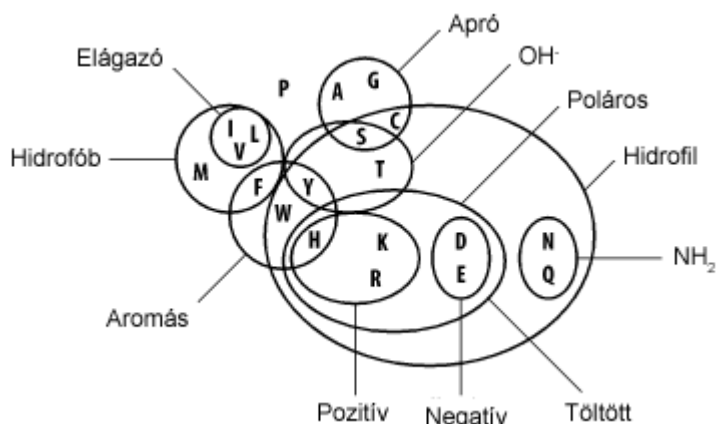
9. ábra. Globális és lokális illesztés.

A globális illesztés klasszikus példája Needleman-Wunsch algoritmus [18], amely megpróbál minden maradékot illeszteni minden szekvenciában. Általában akkor eredményes, amikor az illesztendő szekvenciák közel azonos hosszúságúak. Az algoritmus konkrét pontozási rendszer használ, amely közben a két szekvencia maximális egyezését keresi lehetséges résekkel. A lokális illesztés esetén a Smith-Waterman féle algoritmust [19]. szokás megemlíteni, amely nagyon hasonló az előző algoritmushoz, mindössze három apró módosítás van.

Az itt említésre került módszerek többsége azonban mára a gyakorlatból majdnem teljes mértékben eltűnt, ennek számos oka van. Ezek az algoritmusok bár biológiai szekvenciákkal manipulálnak az egyes szabályok inkább informatikai jellegűek, mint biológiai elgondolások. A pontozási rendszer ezáltal csak közelíti a valóságot. Természetesen az alapváltozatnak léteznek módosított variánsai, amelyek a problémákat igyekeznek kiküszöbölni, az algoritmust gyorsítani. Ugyanakkor ezek algoritmusok elég számításigényesek, nagy adatbázisokban való keresésre nem kifejezetten alkalmasak. Fontos azonban látni, hogy az egyik legpontosabb eredményt a lokális illesztés szolgáltatja, amelyet a mai is széles körűen használt kereső algoritmusok is használnak. (lásd: 3.4 fejezet) Továbbá elmondható, hogy habár minden szekvencia illesztése logikailag elképzelhető, a végeredményben a megfelelő biológiai jelentőség értelmezése a felhasználóra hárul, mert azt az algoritmus nem adja meg.

3.3.2 Szekvencia hasonlóság

Két szekvencia hasonlóságának meghatározásának másik fajta szemlélete annak a felismerése, hogy az biológiai szervezetek felépítésének tanulmányozásával olyan heurisztikák ismerhetők fel, amelyek az egyes szekvenciák változásainak megértésében segítségünkre lehetnek. Az aminosavak az oldalláncuk kémiai tulajdonságaik alapján különböző halmazokba sorolhatóak, amelyek között átfedések vannak. (10. ábra)



10. ábra. Az aminosavak csoportosítása kémiai tulajdonságaik alapján.

Evolúciós szempontból az tűnik logikusnak, hogy az egyes aminosavak a hozzájuk hasonló aminosavakra cserélődnek. Amennyiben az egyes aminosav egy teljesen más aminosavra változik, akkor a fehérje térszerkezete teljes mértékben megváltozhat, amely egyenértékű a fehérje funkcióvesztésével. Az 1960-as évek végén és az 1970-es évek elején Margaret Dayhoff kvantitatív eljárásokat dolgozott ki az aminosavak előfordulásának hasonlóságának megismerésére. A várakozásoknak megfelelően kutatása igazolta a feltételezéseket. Így olyan információk nyerhetők váltak ismertté, hogy az egyes aminosavak milyen aminosavakkal létesítenek nagyobb valószínűséggel kapcsolatot, az egyes szekvenciákban bizonyos mintázatok is felismerhetővé váltak. Margaret továbbá meghatározott egy fogalmat, a *LOD score*-t, amely megadja, hogy mennyi a valószínűsége annak, hogy az egyes aminosavat egy másik helyettesíti. Ismerve a törzsfejlődésre jellemző változásokat ez a folyamat kétirányúan,

asszimmetrikusan jellemezhető. Például A aminosavból nagyobb valószínűséggel lesz B, mint fordítva.

Az elgondolás a hasonlósági mátrixok megjelenéséhez vezetett. Először azonban említsük meg, hogy a hasonlóságok elemzésének ennek a fajta módszerénél az előzőekben említett réseknek is fontos szerepük van. A rések megjelenését nagyobb súllyal kell büntetni (*gap opening penalty*), a rés növekedése (*gap extension penalty*) viszont annyira nem jelentékeny, ez kevésbé ront a jóságon. Az értelmes összerendezéshez figyelembe vesszünk egy mátrixot, amelynek sorait és oszlopait a szervezetekben előforduló aminosavak alkotják. A főátlóban lévő elemek azt jelölik, hogy az egyes aminosavak a két szekvenciában nem változnak. Tekintsük az egyik szekvenciát az oszlopoknak, a másik szekvenciát a soroknak. A mátrix értékei olyan súlyok, amelyek meghatározzák az egyes aminosavak megváltozásának jóságát. Természetesen a legpozitívabb, ha az aminosav nem változik meg, a legnegatívabb, ha az aminosav úgy változik meg, hogy egy teljesen más kémiai tulajdonságú kerül a helyére. Ezeknek az értékeknek a meghatározása a mátrix használhatóságának kulcsa. Gyakorlatilag ezzel a mátrixszal az evolúciót foglaljuk össze tömören.

Széles körűen két pontozó mátrix család terjedt el, az egyik a PAM (*Percent Accepted Mutation*), a másik a BLOSUM (*BLOCKS SUBstitution Matrix*). Ezeket a mátrixokat főleg az 1970-es években alkották meg, amikor a szekvenált fehérjék száma igen alacsony volt. A PAM mátrixot Margaret és munkatársai gondozták, főként arra voltak alkalmasak, hogy bizonyos rövid távú evolúciós fejlődést jellemezzenek, illetve, hogy akkoriban lehetőséget nyújtott nagyobb aminosav-távolságok extrapolálására. A BLOSUM mátrixokat igyekeztek minél több tapasztalat alapján elkészíteni, ezek inkább bizonyos fehér családkból különítettek el csoportokat. Különböző bizonyossági fokkal állapítja meg a mátrix, hogy az a fehérje mennyire tartozik abba a csoportba. Például a BLOSUM62 mátrix (11. ábra) a hasonlóságot 62%-os valószínűséggel állítja. Ezeket a különböző hasonlósági mátrixokat a napjainkban széles körűen elterjedt kereső algoritmusok (például BLAST, FASTA) is használják.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

11. ábra. A BLOSUM62 mátrix

3.4 BLAST

A Basic Local Arrangement Search Tool egy olyan algoritmus, amely az előzőekben bemutatott szekvencia illesztő technikák továbbgondolása, valamint egyben egy olyan programcsomagot jelöl, aminek különböző változatai nukleinsavakat, fehérjéket és ezek átkódolt változatait tartalmazó adatbázisokban keres. A programot 1990-ben publikálták [20]. Azóta számos variációja látott napvilágot, de a rendszer alapjául szolgáló algoritmus változatlan maradt. A BLAST egy heurisztikus keresés, amelyet azzal a szándékkal terveztek, hogy a szintén heurisztikus FASTA keresésnél gyorsabban szolgáltatson eredményt, kevés pontosság feláldozásával. Az eljárás jól kalibrált hasonlósági mátrixok segítségével igyekszik olyan kielégítően közelítő eredményt adni, amelyet egyébként a meghatározott feladatra optimalizált dinamikus programozási algoritmussal tudnánk elérni.

Ahhoz, hogy BLAST programot, vagy az Interneten használható felületeket, programozói interfészeket megfelelően használni tudjuk, ismernünk kell a BLAST algoritmusát, hogy az egyes bemeneti értékek értelmét tisztán lássuk. A keresés alapját három réteg adja, a vetés (*seeding*), a kiterjesztés (*extension*) és a kiértékelés (*evaluation*). [21] Rövid kitérő: mivel a magyar szakirodalomban nem található referencia a BLAST algoritmusának kulcsszavainak magyar vonatkozásával kapcsolatban, ezért a fejezet hátralévő részében a kifejezést először megadom magyarul és angolul, a későbbiekben pedig az eredeti szövegben használt angol kifejezéssel hivatkozom rá. Így az érdeklődő olvasó a szakirodalomban könnyebben utána tud nézni az egyes kifejezéseknek. Visszatérve a BLAST algoritmushoz, úgy érzem, hogy az elhangzott célkitűzések némileg magyarázatra szorulnak. Hogyan lehetséges az, hogy a BLAST rövid idő alatt hatékony keresést hajtson végre egy olyan biológiai struktúrában, amelynek bonyolult változékonyságát figyelembe kell venni? A kulcs az, hogy az algoritmus először meghatározza azt a keresési teret, amiben a keresést elvégzi, és utána itt végzi a szekvencia illesztést. Ellentétben például a Smith-Waterman algoritmussal, amely az egész keresési teret bejárja.

3.4.1 Seeding

Az algoritmus feltételezi, hogy a jelentékeny szekvencia illeszkedéseknek közös szavaik (*words*) vannak. Szavak alatt egy előre meghatározott számú betűk konkatenációját értjük. Az AGTCT szekvenciában, ha a hárombetűs szavakat akarjuk meghatározni, akkor azok: AGT, GTC és TCT. Tehát állandó hosszúsággal minden lehetséges részhalmaza az eredeti szekvenciának. Az algoritmus először meghatározza a kereső szekvenciának és a cél szekvenciának közös szavait, ezt szótalálatoknak (*word hits*) nevezzük. A keresési mezőt azok a régiók alkotják, ahol ilyen találatok előfordulnak. Értelmszerűen az adott alkalmazástól függően a szavak hosszát jól kell megválasztanunk, mert túlságosan kis szóhossz esetén a keresés nagyon sokáig tarthat, túlságosan nagy szóhosszúság esetén pedig értékes találatokat veszíthetünk el. Nukleotid adatbázisokban történő keresés során a szavak hossza tizenegy betű ($w = 11$). Előfordulhat, hogy két szekvencia nem teljesen egyezik meg, ezért az algoritmus egy szomszédsági értékkel (*neighborhood*) dolgozik, amit T jelöl. Ez azt jelenti, hogy az egyes szóhosszhoz generálunk egy olyan táblázatot hasonlósági mátrix (például BLOSUM62, PAM200) segítségével, amely az egyes ábécé beli eltéréseket különböző

pontokkal jellemez. Minél nagyobb a T érték, annál nagyobb a hasonlóság. A T értékek meghatározásában fontos faktor az eltérések érzékenysége és az algoritmus futásának sebessége.

A keresési tér szűkítése ezek után a *two-hit* algoritmus segítségével történik. Az algoritmus olyan találatokra koncentrál, ahol a *seed* pontok valamilyen összefüggést mutatnak (hasonlóan a *dot-matrix* módszerhez). Az egymáshoz közel álló több pontot nagyobb valószínűséggel tekint találatnak, mint az elszigetelteteket.

3.4.2 Extension

A kiterjesztés folyamata során rendelkezünk olyan pontokkal, amelyekkel érdemes foglalkoznunk. A kiterjesztés során a pont bal és jobb irányába vizsgálódva egyre több betűt választunk ki az adatbázisban tárolt szekvenciából, összehasonlítva a kereső kifejezéssel. Ez akár a végtelenségig is tarthat, ebben az esetben az egész adatbázist illesztenénk, ami a futási idő szempontjából eléggé kritikus. Tehát az algoritmusnak meg kell tudni határoznia, hogy mi az a határ, ami után nem növeli a kiválasztott pontot, ezáltal kijelölve azt a szekvencia részletet, amelyet hasonlónak tekint. Két számlálót dolgozunk, az egyik a pontszám (*score*), a másik az eldobási pontszám (*drop off score*). A *score* a pontozási rendszer szerint növekszik vagy csökken. Szokás X -szel jelölni a *drop off score*-t, amelynek értéke akkor növekszik megadott ponttal, amikor eltérést tapasztalunk, azaz büntetünk (*penalty*). Ez az érték azonban csökkenhet, amikor a rá következő betű hasonlóságot mutat. A *seed* növelését addig végezzük, ameddig az X értéke el nem éri a küszöbértéket. Kis X esetén nagyobb az esélye, hogy homológ szekvenciát találunk, de kevesebb találatunk lesz. Nagy X esetén több találatunk lesz, de nem biztos, hogy azok relevánsak lesznek. Fontos megjegyezni, hogy ezek a fajta elgondolások, próbák akár statisztikailag stabil eredményt adhatnak, azonban a biológiában egy aminosav eltérés is igen erőteljes jelentéssel bírhat. Az egyszerűség kedvéért egy olyan példával szemléltetném a megállási algoritmust, amely a hasonlóságra +1 pontot, az eltérésre -1 pontot ad, nem kezeli a réseket, valamint $X=5$.

```
A szeles réten repked a madár a magasban.
A szoros téren repked a bagoly unottan.
1 232101 01012 345678 9 898765 4          <- score
0 001232 32321 000000 0 101234 5          <- drop off score
```

Látható, hogy a pontszám számláló értéke ingadozik, majd mondat alanyának (madár, bagoly) a második karakternél felveszi a lokális maximum értéket, a kilencet, ezek után folyamatosan csökken. A X értéke ingadozik, de nem éri el azt küszöbszintet, amit meghatároztunk, csak a mondat vége előtt, amikor elkezd nagymértékben különbözni a két szekvencia. Amikor eléri az ötöt, az algoritmus megáll, nem vizsgálja tovább a szekvenciákat. A *score* számlálóban megkeresi a legutolsó legnagyobb értéket és addig tartja meg a szekvenciát. Jelen esetben „A szeles téren repked a ma”, illetve „A szoros téren repked a ba”. A 12. ábrán a pontszám számláló értékének változását szemlélteti, illetve a vágás meghatározását a lokális szélsőérték alapján.

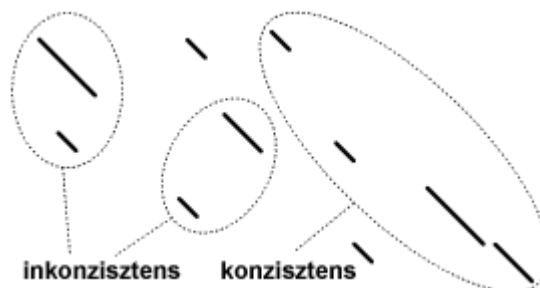


12. ábra. A pontszám változása a szekvencia kiterjesztés során és a vágási pont meghatározása

Természetesen a gyakorlatban alkalmazott BLAST a dinamikus programozással összefüggésbe hozható, a példában bemutatottnál összetettebb algoritmussal rendelkezik és a réseket is figyelembe veszi. A *drop off score* számlálót a rések nyitása és növelése is befolyásolja. A *score* számolásának szabályai az egyes alkalmazástól függően változnak, például nukleinsav adatbázisok esetén a hasonlóság +5, a különbség -4 pontot ér.

3.4.3 Evaluation

Amikor *seed* pontok illesztése befejeződött, létrejöttek a hasonló szekvenciák, a következő lépés ezeknek a szekvenciáknak a statisztikai szignifikanciájának meghatározása. A szignifikáns szekvenciákat *HSP*-nek szokták nevezni. A szignifikanciának meghatározása alapvetően statisztikailag könnyű. Erre feladatra az illeszkedési küszöbértéket (*alignment threshold*) használja az algoritmus, amelyet S jelöl. Az eredményeket növekvő sorrendbe rendezve, S küszöbérték felett az eredményeket a próba elfogad, az alatt pedig elveti. A már megszokott módon ennek értékét körültekintően kell megválasztani, mert túlságosan kis érték esetén nem releváns adatokat is elfogad a próba, túlságosan nagy érték esetén valódi találatokat veszíthetünk el. Problémát jelentenek a találatok biológiai értelmezései is, mert az egyes HSP találatok egy közös találatot is jelenthetnek. Amennyiben több HSP egy konzisztens vonal intervallum darabkáit adja meg kisebb-nagyobb résekkel, azokat konzisztens találatoknak nevezzük. (10. ábra)



13. ábra. Konzisztens és inkonzisztens találatok szemléltetése a keresési térben

Biológiai szemszögből ez azt jelenti, hogy a szekvencia végén a kódolt fehérje az 5' irányú végén N terminálusra számítunk, a 3' irányú végén C terminálusra és nem fordítva. Ezért az algoritmus a meglévő HSP-eket megpróbálja konzisztens és

inkonzisztens találatokba szervezni. A konzisztens találatok esetén figyelembe veszi az átfedéseket, ismétléseket. Amikor a találatok konzisztens csoportokba szervezése befejeződött a találatokat egy végleges küszöbérték (*final threshold*) segítségével még egyszer szűri. Ezt a küszöbértéket az egész keresési mező tulajdonságainak és az E értékének figyelembevételével végzi. Az E , amely a Karlin-Altschul egyenletből (1) következik, értéke minél nagyobb, annál nagyobb valószínűséggel fogad el a próba véletlen szekvenciákat, azaz az egyes homológiák annál nagyobb valószínűséggel a véletlen művei.

$$E = kmne^{-\lambda S} \quad (1)$$

Azaz E (Expect) az a függvény, amely adatbázisban történő keresés során illeszkedések számának elvárt értékét adja meg, k egy konstans, m a keresési szekvenciában szereplő betűk száma, n az adatbázisban szereplő betűk száma, azaz $m*n$ a keresési mező mérete, $-\lambda S$ normalizált eredmény (hasonlósági mátrixokból) szorzataként.

A másik fontos paraméter, amely a gyakorlatban használatos a lekérdezések során a P érték (3), amely az E értékből számolható (2), és azt fejezi ki, hogy bizonyos illeszkedésre milyen gyakran lehet számítani.

$$E = -\ln(1 - P) \quad (2)$$

$$P = 1 - e^{-E} \quad (3)$$

Gyakorlati szempontból elmondható, hogy, amennyiben $E < 0,01$, az már egy olyan küszöbszint, amely egy biológiailag releváns homológiát feltételez. Továbbá az egyenletekből következik, hogy 0,001-nél kisebb értékek esetén E és P lényegében megegyezik.

3.5 Genomikai adatbázisok, szolgáltatások

A közel egy évtizede elkezdődött és napjainkban is zajló genom szekvenálási projektek során (HGP: *Human Genome Project* 1990-2003, MGP: *Microbial Genome Program*, növényi genom programok), rövid időn belül hatalmas adatmennyiség keletkezett, mely különböző nemzetközi adatbázisokban került eltárolásra. Ezen adatbázisok adatmennyiségének feldolgozása a munka méreteiből adódóan csak informatikai eszközökkel kivitelezhető. A rendelkezésre álló adatbányászati eszközök, valamint biológiai tudás ötvöztetésével ezekből az adathalmazokból olyan információ kiszűrésére lenne igény, mely segítségével a gyógyszerkutatások hatékonysága megnövelhető. Közel húsz éve alakultak a nagyobb adatgyűjtő helyek amelyek a hatalmas mennyiségű adathalmaz tárolásra szolgálnak. Ezek a nyilvános adatbázisok bárki számára hozzáférhetőek online és letöltött formában offline módon. Nem az összes ismert szekvencia nyilvános, vannak hozzá nem férhető adatok is.

Online adatbázisok:

- EMBL (European Molecular Laboratory) (Európa- Anglia)

- Genbank (USA)
- DDBJ (Japán)
- EBI (European Bioinformatics Institute)
- EMBnet
- Sanger Institute
- NCBI (National Center of Biotechnology information)
- TrEMBL (DNS adatbázis transláció)
- PUBMED

Az Internetes keresőfelületek böngésző általi használatán kívül programozási interfészeken keresztül is van lehetőség megközelíteni ezeket az adatbázisokat. Számos online adatbázis megfelelő interfészt biztosít az adatok lekérdezésére. Ezeket a hozzáférési lehetőségeket még a mostanában divatos Web Service által meghatározott szabvány megjelenése előtt kezdték el fejleszteni. Így az egyes interfészek speciális illesztő programok írását követelik meg, de természetesen léteznek szabványos SOAP (*Simple Object Access Protocol*) megoldások is.

3.5.1 NCBI Entrez Programming Utilities

Az NCBI az Entrez-ben található adatbázisokhoz való kereséshez és adatok lekérdezéséhez és beküldéséhez egy viszonylag jól dokumentált lekérdezési felületet nyújt. [22] A szolgáltatás alapvetően URL lekérdezések segítségével elérhető, de létezik egy fejlesztés alatt álló SOAP megoldás is. Az előbbit inkább a script nyelvek használják könnyedén, az utóbbit a szabványosítás és a támogatás megjelenése miatt a robusztusabb fejlesztői környezetek, például a Java és a .Net keretrendszer támogatja.

Többféle eszköz található a honlapon, egyszerű kereséshez az ESearch és az EFetch használata szükséges. Az ESearch segítségével egy adott adatbázisban kereshetünk egy megadott kulcsszóval. A visszaadott XML válasz tartalmazza a találatok Entrez azonosítóit. Az EFetch segítségével az egyes azonosítók alapján a bejegyzések részletes adatait, különböző formátumokban (például: FASTA, GenBank, egyszerű szöveg, XML) kérdezhetjük le.

3.5.2 NCBI BLAST Web Service

Az előzőekben részletesen tárgyalt BLAST algoritmust (lásd 3.4 fejezet) használó programcsalád az NCBI honlapján keresztül használható. Azonban, ha a szolgáltatást egy külső programból szeretnénk használni, akkor a web szolgáltatást kell igénybe vennünk. [23] A szolgáltatás nem rendelkezik fejlett dokumentációval, a szolgáltatást bemutató leírás vázlatos, kevésbé szemléletes. Azonban elérhető egy Perl nyelven írt példaprogram, amely a használatot szemlélteti. A lekérdezést itt is HTTP (*Hypertext Transfer Protocol*) kérés segítségével hajthatjuk végre, ahol az URL-ben kell a különböző paramétereket átadnunk. A visszaadott válasz formátuma ASN.1 (*Abstract Syntax Notation One*), XML (*eXtensible Markup Language*) vagy egyszerűsített táblázat lehet.

3.5.3 BioMart

A BioMart egy lekérdezés orientált adat menedzselő szoftver, amelyet az EBI (*European Bioinformatics Institute*) és a CSHL (*Cold Spring Harbor Laboratory*) munkatársai fejlesztenek. [24] A rendszer különböző adatbázisok integrálása alkalmas, az adatbázisok közötti összefüggések felhasználásával olyan lekérdezések fogalmazhatóak meg, amelyek különböző adatbázisokat érintenek. Mindezt a felhasználó számára transzparens módon végzi. A szoftver adatbányászati célokra alkalmazható, nagy mennyiségű adat nyerhető ki segítségével különböző adatbázisokból. Két fajta felületet biztosít, egy webes lekérdező felületet (*Martview*, 14. ábra) és egy programozói interfészt (*Martservice*). A webes felületen programozási ismeretek nélkül van lehetőségünk a lekérdezés összeállítására. Az eredményeket a rendszer táblázatos formában adja meg. Alapvetően gén specifikus információk kérdezhetőek le, úgy, mint szekvenciák, homológiák, adatbázis specifikus azonosítók.

14. ábra. Martview felület.

A lekérdezés menete:

1. A lekérdezés során először az érdeklődés területét, a *DataSet*-t kell kiválasztani, és ezen belül egy speciális adatbázist. Ez például lehet az Ensembl gén adathalmaz, amelyből az emberre vonatkozó adatbázist választjuk ki.
2. Következő lépésként meg kell határoznunk, hogy milyen bemeneti információkkal rendelkezünk, ez a *Filters*. Az előző példa gondolatmenetét folytatva, itt például emberi gének HGNC azonosítóit adhatjuk meg.
3. Végül meg kell határoznunk, hogy milyen információkra vagyunk kíváncsiak. Az *Attributes* menüpontnál adhatjuk meg ezeket a beállításokat. A génekkel kapcsolatban különböző tulajdonságokat, homológokat, szekvenciákat, SNP-eket

kérdezhetünk le. A fentebb tárgyalt példa esetében például beállíthatjuk, hogy a az emberben megadott gének egérben található homológ génjeit szeretnénk lekérdezni, kimenetként az egér gének MGI azonosítóival.

Természetesen ennél a példánál jóval összetettebb kérdések is megfogalmazhatóak. Szoftveres hozzáférés esetén a rendszer különböző API-kal rendelkezik. Jól használható a BioPerlbe épülő csomag, létezik illesztő felület Java-hoz, R-hez. Ezeken a kommunikációs csatornákon keresztül a rendszer magas szintű biológiai lekérdező programokba is képes betagozódni, mint például a *Taverna*-ba és a *Galaxy*-be. Elérhető egy Web Service csatolófelület is, amely nem a szabványos WSDL által leírt kommunikációt valósítja meg, hanem egy speciális, de könnyen átlátható XML segítségével hajthatunk végre lekérdezéseket. Az egyes lekérdezések összeállítását után lehetőség van a lekérdezés XML verziójának megtekintésében, ami egy saját illesztőprogram tervezésében nyújthat segítséget.

4. Feladatspecifikáció

A dolgozat által megvalósítandó szoftver célja egy olyan webes vagy ablakos alkalmazás létrehozása, amely integrált felhasználói felülettel rendelkezik, egy adott munkafolyamatot valósít meg különböző bioinformatikai adatbázisokban való kereséssel.

A program egy előre meghatározott adatbányászati folyamatot kell, hogy megvalósítson, amely során bioinformatikai adatbázisokkal és szolgáltatásokból adatokat kérdez le, a válaszként kapott adatokat elemzi és a munkafolyamat szempontjából releváns adatokat egy lokális relációs adatbázisban tárolja. A rendszer kialakítása során törekedni kell arra, hogy a munkafolyamat a futtató állományoktól elkülönülten helyezkedjen el, hogy a program forráskódjának átírása nélkül a munkafolyamat módosítható legyen. Továbbá az egyes külső adatbázisokhoz, szolgáltatásokhoz való kapcsolódás transzparens, jól konfigurálható módon történjék.

A munkafolyamat célja olyan gének, nukleinsav szekvenciák keresése, amelyek a megadott kulcsszó esetén releváns kutatási eredményekkel rendelkeznek az ember és a kiválasztott modellállat vonatkozásában. A keresés eredményét olyan, az emberben előforduló gének és nukleinsav szekvenciák alkotják, amelyekkel kapcsolatban tudományos hivatkozás az adott kulcsszóra vonatkozóan nem található az NCBI genom adatbázisában. Tehát a cél olyan új, feltételezhetőleg ismert, de még nem kutatott gének és DNS szekvenciák keresése, amelyek a kulcsszóval kapcsolatban egy esetleges további kutatás alapját képezhetik.

A munkafolyamat első lépéseként a szoftver az NCBI genom adatbázisában keres a megadott kulcsszóval. A visszakapott eredmények halmazából kell, hogy kiválogassa a keresésnek megfelelő találatokat. Az egyes találatok közötti homológiát a rendszernek fel kell ismernie, hogy a találatok szűrését elvégezhesse, és ezáltal folyamat végeredménye láthatóvá váljon. A homológiák felismerésére a BioMart szolgáltatás és a BLAST használata ajánlott. A keresőszavak alapvetően olyan kulcsszavak, amelyek valamilyen multifaktoriális, poligénus betegségek (*lásd: 2.6.2.*) nevei.

A rendszernek rendelkezzen grafikus, intelligens, integrált felhasználói interfésszel, amely a keresések indítását, nyomon követését és a végeredmény megtekintését teszi

lehetővé. Az elvárás szerint a keresések indítását egy űrlap segítségével lehet végrehajtani, aminek kulcs eleme a keresendő kifejezés.

5. A megvalósítás lehetséges módjai

A feladat-specifikációban szereplő igények kielégítése általános célú alkalmazásokkal nem megvalósítható, a megoldáshoz összetett céleszközre van szükség. Ezen eszköz kiválasztásakor vagy tervezésekor, különböző szempontok szerinti mérlegelésre van szükség, hogy a rendszer minden igényt kielégítsen.

Ilyen szempontok például:

- a megvalósításhoz használt architektúra
- felhasználói felület felhasználó barátsága
- a munkafolyamat leírásának, felépítésének egyszerűsége
- a futtatási környezet igényei
- a rendszer kliens-szerver megvalósítása
- több felhasználó támogatása
- a munkafolyamat futási idejének optimalizálása

Látható, hogy ezek a szempontok bizonyos kérdéscsoportok köré szerveződnek. Az első ilyen szempont a munkafolyamattal kapcsolatos. Egy újonnan írt programban a munkafolyamatot egy beépített algoritmus is megvalósíthatja, vagy magasabb szinten egy olyan munkafolyamat leíró értelmezőt kell alkotni, amely egy bizonyos leíró nyelvet értelmez, és a beolvasott információk alapján hajtja végre magát munkafolyamatot. Másik lehetséges választás, ha egy olyan meglévő rendszert használunk, amelyet munkafolyamatok kezelésére fejlesztettek ki. Ez a fajta döntés a többi paramétert meghatározza, amelynek részletes elemzését az alábbiakban adom meg, de előtte néhány szót ejtek a feladatban megvalósítandó munkafolyamatról.

5.1 Munkafolyamat

A szoftver által megvalósítandó folyamat jól elkülöníthető egységekből épül fel, amelyek között az alapvető függőségek jól láthatóak. Alapvetően egy olyan „fekete doboz” transzformációról van szó, amely meghatározott bemenetekkel rendelkezik és meghatározott formátumú kimenetet vár. A doboz belsejében megvalósított folyamat ebben az esetben egy olyan adattranszformáció, amelynek kritikus lépéseit a külső adatbázisokkal történő interakció szolgáltatja. Ezen lekérdezések köré alapvetően helyi adatmentések társulnak, illetve szükség van a lekérdezett adatok rendszerezésére, szűrésére. A már előzőekben ismertetett algoritmus ábrázolására logikus választásnak egy összetett folyamatára tűnik, amely rendelkezik párhuzamos végrehajtással, illetve a párhuzamosságból visszatérve szinkronizációs ponttal. Ennek a fajta leírásnak leginkább a strukturált, szinkronizáló összefésülés (*Structured Synchronizing Merge*) folyamatára felel meg. [25] A megvalósítandó folyamat ugyanis egy kiindulási kereséssel kezdődik, amelyet párhuzamosítható keresések és kiértékelések követnek,

amelyet egy olyan összerendezés zár le, amely csak akkor hajtható végre, ha mindkét párhuzamos szál futása befejeződött, tehát az összerendezés előtt egy szinkronizációnak kell lennie. Természetesen ez a fajta folyamatábra ekvivalenssé tehető egy teljesen szekvenciális adatfeldolgozási folyamatábrával, amelyben az egyes párhuzamos szálakat egymás után illesztjük.

5.2 Magas szintű folyamatleírók, elosztott rendszerek

Számos olyan magas szintű folyamatleíró projekt létezik, amely segítségével a dolgozatban tárgyalt lekérdezési feladatot megoldhatjuk. Ezek általában olyan projektek, amelyeket tudományos számítások elvégzésére fejlesztenek, és a felhasználó számára transzparenssé teszik az egyes szolgáltatások elérését és a futtató környezetet. Ilyen rendszerek például: *Taverna* [26], *Kepler* [27], *Pegasys* [32], *Triana* [33]. A dolgozat keretein belül nagyobb figyelmet a Tavernának szentelek. A Pegasys és a Triana egyetemek által fejlesztett bioinformatikai leíró rendszer, amelyet főleg microarray génchip genom adatok feldolgozására fejlesztettek, úgy, hogy a szoftver elosztott, grid rendszereken legyen képes futni. Ezeknek a projekteknek a fejlesztése néhány éve szünetel. A Kepler elsősorban nem csak bioinformatikai adatok kezelésére kifejlesztett rendszer, számos más tudományos munkafolyamat megvalósítására is alkalmas. Azonban a program robosztusságából fakadóan a használata is meglehetősen bonyolult.

Mielőtt a Taverna ismertetésére rátérnék, röviden összefoglalnám, hogy milyen érvek sorakoznak *pro et contra* egy ilyen rendszer használata mellett.

Pro:

- a workflow könnyen, vizuálisan szerkeszthető
- az egyes illesztőprogramokat a rendszer beépítetten kínálja
- futtatás grid rendszereken, ezáltal nagy számítási kapacitás várható
- meglévő fejlesztés, programozást alapvetően nem igényel

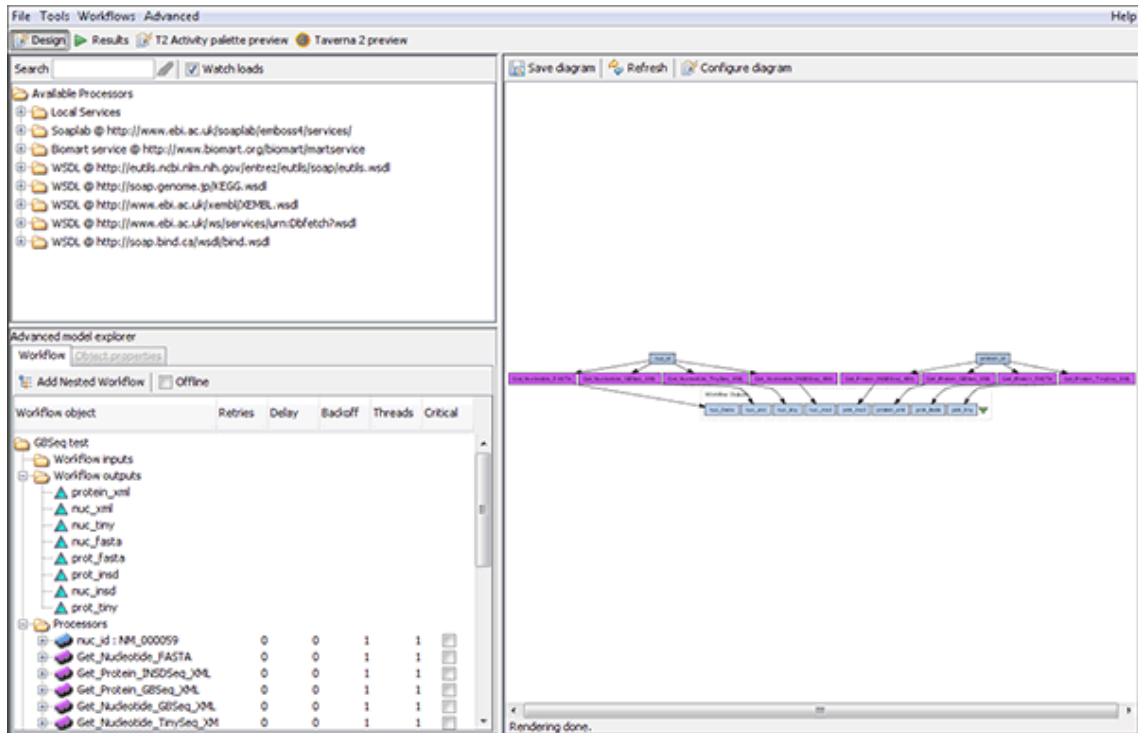
Contra:

- grid rendszereken történő futtatás meglehetősen bonyolult
- többfelhasználós rendszer kialakítása alapértelmezés szerint nem támogatott
- a végeredmények vizualizációja erősen korlátozott
- speciális igények esetén meglévő az ilyen rendszerek továbbfejlesztése bonyolult lehet

5.2.1 Tapasztalatok a Taverna rendszer használatával kapcsolatban

A Taverna projekt a myGrid komponens család keretein belül fejlesztett szoftver. A myGrid egy összefoglaló projekt, amely különböző tudományos eszközöket kíván nyújtani nyílt forráskódú alapokon. A Taverna egy olyan workflow tervező és végrehajtó szoftver, amelyet kifejezetten bioinformatikai adatbázisokhoz terveztek. A programot Java nyelven fejlesztik, ami lehetővé teszi, hogy a program platform független alkalmazás legyen. A szoftver telepítése egy vékony kliens letöltéséből és annak elindításából áll. Első futtatáskor letölti a szükséges beépülő modulokat, amelyek a különböző web szervizeket kezelik. Alapértelmezés szerint a program a nagyobb

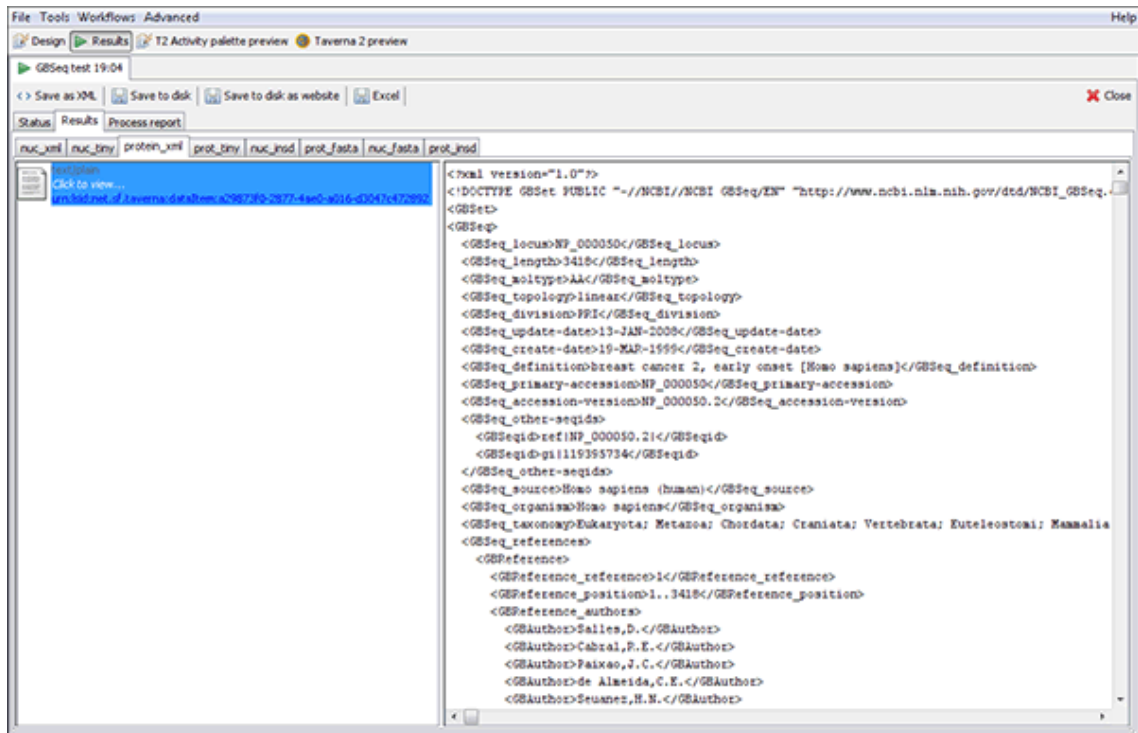
bioinformatikai adatbázisok (NCBI, EBI, GenomeNet) web szervizei elérhetőek, illetve a BioMart szolgáltatás.



15. ábra. A Taverna munkafolyamat tervezői felülete

A felhasználói felület (15. ábra) egyszerű, alapvetően két fajta nézet között választhatunk. Van egy tervezési és egy végrehajtási nézet. A tervezési nézetben lehetőségünk van, az egyes SOAP szolgáltatások, mint végrehajtási egységek összeillesztésére, vezérlésére. Az egyes komponensek közötti szekvencia, függőségek megadhatóak, bár a különböző adatstruktúrák közötti konverziók nehézkesek. Ezeket olyan beépített lokális végrehajtó egységekkel kívánja a program megoldani, mint például beépített Iterator, Enumerator, sztring függvények, stb. A workflow beágyazva tartalmazhat más workflow-kat, így az egyes elemek akár más összetett munkafolyamatok is lehetnek, ami az újrahasznosíthatóság és az átláthatóság szempontjából előnyös. Mindezekből jól látszik, hogy a Taverna jól használható kezdeményezés, amely segítségével bioinformatikai workflow-k könnyen összeállíthatóak, némi informatikai ismerettel. Az elemzés teljességének érdekében a hátrányokról is szót kell ejteni. A program felhasználói felületének fejlettsége a manapság megszokott kényelmi szolgáltatások töredékével rendelkezik. Az eszköztárak, felbukkanó ablakok sok helyen nehézkesen találhatók meg, így az egyes munkafolyamatok összeállítása eleinte sok fejtörést okozhat. Legnagyobb hátrányként azt említeném meg, hogy a workflow diagram nem szerkeszthető, az egyes egységek helyben nem szerkeszthetőek, nincs *drag&drop*, az egyes relációkat sem kezeli a rendszer a diagramon. Ezáltal a diagram egy vizuális visszacsatolássá degradálódik és elveszíti azt a fajta élményt, amely a felhasználó szemszögéből a kényelmességet, logikus felhasználást jelentené. A másik hátrány a bemenő adatok kezelésének problémája. A legegyszerűbb módon az inputok sztring konstansként vannak jelen a

rendszerben. Ennél valamivel kényelmesebb megoldást kínál a külső, szöveges fájlból történő beolvasás, azonban ez sem nyújt felhasználói felületet a bemenő adatok vizualizációjához.



16. ábra. Eredmények megjelenítése a Taverna végrehajtási nézetében

A végrehajtási nézetben (16. ábra) a rendszer az általunk összeállított munkafolyamatot futtatja. A végrehajtás közben figyelemmel kísérhetjük az egyes lépések aktuális állapotát. A folyamat legvégén az egyes kimenetek eredményeit tekinthetjük meg, illetve a futással kapcsolatos információk (futási idő, állapot) is elérhetőek. A kimeneteket különböző fájlformátumokba van lehetőségünk menteni. A rendszer alapértelmezés szerint a saját számítógépünkön fut, a folyamat végrehajtása annak kapacitásától és a webszervizek válaszainak sebességétől függ. Elképzelhető elosztott rendszereken való futtatás, de ennek transzparens módon kell történnie. Az egyes szervizek különböző csomópontokba történő kihelyezésére nincs lehetőség, a szoftvert nem ilyen jelleggel tervezték.

Összegzésként elmondható, hogy a Taverna egy jól használható bioinformatikai eszköz, amelyet olyan biológus, genetikus kutatók számára fejlesztenek, akik rendelkeznek informatikai ismeretekkel. Számukra a szoftver használata olyan eszközt jelent, amelynek használatával a munkájuk egyszerűbbé, hatékonyabbá tehető. A rendszer törekszik a már meglévő szolgáltatásokat kihasználva egy egyszerű tervező és végrehajtási felületet biztosítani. A rendszer filozófiájából következően a program nem támogatja a csoportmunkát, az elosztott rendszerek használatát, az eredmények szemléletes vizualizációját.

5.3 Saját rendszer fejlesztése

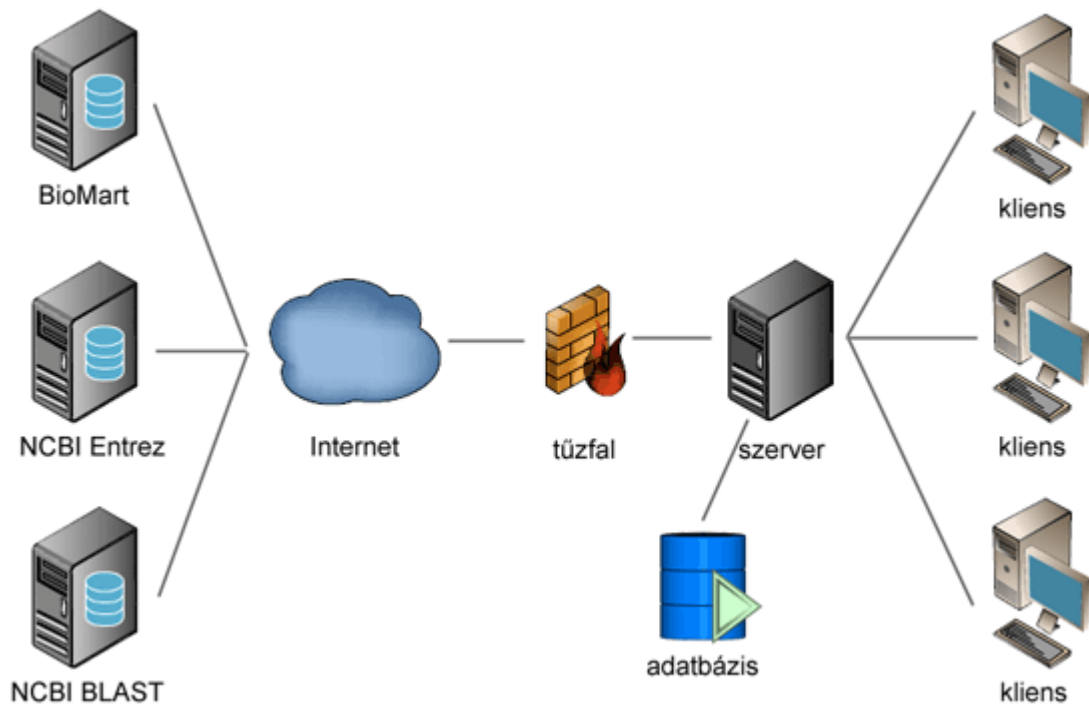
A másik lehetséges megvalósítási alternatívát egy önálló program írása jelenti, amely a specifikációban kitűzött igényeket elégíti ki. Ennek a megoldásnak is természetesen számos előnye és hátránya van, mint ez előzőekben tárgyalt meglévő rendszerek használatának. Az első kérdés, amely a tervezéssel kapcsolatban felmerül, hogy milyen környezetben célszerű a rendszert implementálni, amelyben viszonylag egyszerűen megvalósítható a különböző külső adatbázisokhoz való kapcsolódás, a csoportmunka támogatása, az integrált grafikus felhasználói felület biztosítása, a workflow támogatása. Ezen szempontok figyelembevételével kézenfekvőnek tűnik egy webes szerver-kliens megoldás. A szerveren futó alkalmazás bármikor központilag módosíthatóvá válik, az egyes külső adatbázisok elérése a szerver Internet kapcsolatával megoldható. A lokális adatbázis elhelyezése is a szerveren célszerű, így biztosítva a megosztott tudást, valamint portabilitást. A manapság elterjedt vékonykliensek, a web böngészők minden számítógépen megtalálhatóak, folyamatosan fejlesztik őket, olyan egyre fejlettebb platform független interfészek létrehozását biztosítják, ami magas felhasználói élményt biztosít. A megvalósítás eszköze lehet többek között Java, .Net, Perl/BioPerl, PHP alapú architektúrák alkalmazása. Természetesen más fejlesztői környezetek is számításba jöhetnek, de ezeknek napjainkban ezeknek a rendszereknek a legnagyobb a támogatottsága. Bár számos előnyét láthatjuk ennek az alternatívának, meg kell említeni egy ilyen „from scratch” fejlesztés hátrányait is. Az újonnan készített szoftver nem rendelkezik olyan már meglévő erőforrásokkal, mint a már meglévő rendszerek, ezeket a szolgáltatásokat magunknak kell megterveznünk, implementálnunk. Ez a probléma lényegében a grideken való futtathatóságot és a munkafolyamat kezelő alrendszer fejlesztését jelenti.

6. Rendszerterv és rendszerműködés

A dolgozat keretein belül megvalósított szoftver, egy önálló alkalmazás fejlesztését tűzte ki céljául. Az előzőekben bemutatott lehetséges alternatívák közül azért választottam ezt a fajta megoldást, mert a rendelkezésre álló meglévő rendszerek továbbfejlesztése túlságosan sok erőforrást igényel egy egyedi rendszer fejlesztéséhez képest. Azaz, megfelelő technológiák alkalmazásával az elérni kívánt cél egy új alkalmazás fejlesztésével egy olyan új rendszer hozható létre, amely a megadott problémákat hatékonyabban oldja meg. Ez alatt elsősorban a szerver-kliens architektúrát, a felhasználók közötti kommunikációt, a hordozhatóságot és a továbbfejleszthetőséget értem. Továbbá a döntés része az, hogy egy olyan programot kívántam megalkotni, amelynek tervezése és fejlesztése során már meg lévő ismereteimet felhasználva az ismeretek egy új szintjére lépve egy olyan új megoldás alapjait helyezzem el, amely a jövőben további fejlesztések kiindulópontjául szolgálhat.

6.1 Architektúra

A szoftver egy központi webserveren fut. A kliensek a webszervert HTTP kéréseken keresztül szólítják meg, ami az egyes kéréseket átadja a programnak. A rendszer az adatokat a webszerveren egy adatbázis kezelő rendszerben tárolja. (17. ábra) Természetesen a webszerver és az adatbázis kezelő különválasztható, külön szerveren is elhelyezhető. A külső bioinformatikai adatbázisokkal történő kommunikáció HTTP kérések és válaszok formájában történik, az Interneten keresztül. A kliensek és a szerver elhelyezése többféleképpen konfigurálható. A fejlesztői környezet esetében a szerver és a kliens lehet ugyanazon a számítógépen. A program működő verziója során az egyes kliensek vagy ugyanazon hálózaton helyezkednek el, mint a szerver (LAN), vagy az Interneten (WAN) keresztül kommunikálnak a szerverrel. Ez lehetőséget biztosít a manapság gyakori hálózati infrastruktúrákba való integrálásra.



17. ábra. Kliens-szerver architektúra külső Internetes adatbázisok elérésével

A szoftver futtatásához Internet kapcsolat és az alábbi programok szükségesek, amelyek mindegyike nyílt forráskódúak. Az egyes elemek a szabványoknak köszönhetően más elemekkel helyettesíthetők, kivéve a szkript interpretert.

- Szkript interpreter:
PHP 5.2.x vagy újabb (PHP 5 vonal támogatott) [28]
- Webszerver:
Apache (Apache 1.3.x, 2.0.x, 2.2.x is megfelelő) [29]

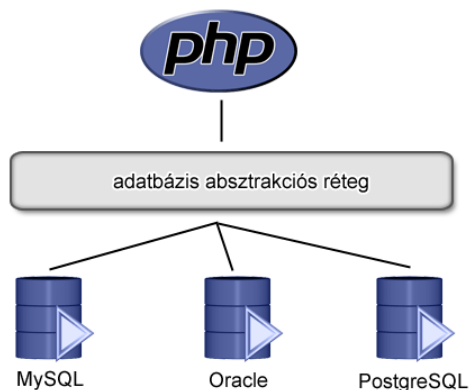
- Adatbázis kezelő
MySQL 5.x (MySQL 5.0 vagy újabb verzió) [30]
- Böngésző
Firefox 2.0 (Más, szabványokat támogató böngésző is alkalmazható) [31]

6.2 Komponensek

A rendszer tervezése során arra törekedtem, hogy az egyes logikailag különálló egységeket komponensekbe szervezzem, amelyek jól behatárolt interfészekkel rendelkeznek, különböző más modulokkal való kommunikációjuk egyértelmű. Ez a fajta szemlélet biztosítja, hogy a komponensek belsejének megváltozása (például egy új adatbázis rendszer használata, új bioinformatikai illesztő program létrehozása) ne vonja magával a többi komponens módosítását. Ugyanakkor a jól deklarált kimeneti és bemeneti pontoknak köszönhetően a komponensek egymással jól tudnak kommunikálni. A program architektúrájából fakadóan néhány komponens szerviz jelleget ölt, ez azt jelenti, hogy singleton tervezési mintát követnek, ami biztosítja, hogy a szervizeket a program bármelyik részében el lehessen érni, használni lehessen. A jelen megvalósítás adatbázis és naplózó szervizt tartalmaz. Az egyes komponensek forráskódjának vázlatát (osztály szintű attribútumok, metódusok) a PHPDoc [34] által ajánlott formátumban adom a meg rövid leírással együtt. Tekintettel arra, hogy a PHP gyengén típusos nyelv, a bemeneti és visszatérési értékek típusát a szintén a komment részek tartalmazzák.

6.2.1 Adatbázis absztrakciós réteg

A program a PHP nyelv OOP sajátosságainak előnyeit igyekszik kihasználni. Ezért létrehoztam egy olyan adatbázis absztrakciós réteget (*Database Abstraction Layer*), amely az adatbázis motortól függetlenül tudja az egyes műveleteket végrehajtani. A jelenlegi megvalósítás során a program MySQL 5.0-hoz kapcsolódik és az adatokat abban tárolja, de kis módosítással más fejlett relációs adatbázis kezelő rendszerhez (például: PostgreSQL, Oracle) is könnyen illeszthető a program (18. ábra). Az adatszervíz szolgáltatás statikus osztályként, singleton tervezési minta szerint van implementálva, így a program bármelyik állapotában egyszerűen és transzparensszerűen elérhető.



18. ábra. Adatbázis absztrakciós réteg

```

Class DB
{
/**
 * Singleton példányt tartalmazó változó
 * @var object
 */
private static $instance;

/**
 * Adatbázis kapcsolat erőforrást reprezentáló változó
 * @var resource
 */
private static $link;

/**
 * Adatbázis név
 * @var string
 */
private static $database;

/**
 * Utolsó lekérdezés
 * @var string
 */
private static $lastQuery;

/**
 * Singleton tervezési mintának megfelelő konstruktor
 * @return void
 */
private function __construct()

/**
 * Kapcsolódás adatbázis kezelő rendszerhez, a megadott adatbázis névvel
 * @param string $host
 * @param string $user
 * @param string $password
 * @param string $database
 * @param integer $port
 * @return void;
 */
public static function connect($host, $user, $password, $database,
                               $port = null)

/**
 * Bontja az aktuális kapcsolatot
 * @return boolean
 */
public static function close()

/**
 * SQL utasítás végrehajtása (pl.: lekérdezés, azaz SELECT)
 * @param string $sql
 * @return Database_ResultSet A visszatérési érték egy Database_ResultSet
 * objektum
 */
public static function executeQuery($sql)

/**
 * SQL utasítást végrehajtása
 * @param string $sql
 * @return integer Az utolsó létrehozott rekord azonosítója
 */
public static function executeInsert($sql)

/**
 * Törölő, módosító SQL utasítás végrehajtása
 * @param string $sql
 * @return integer Az utasítás által érintett rekordok száma
 */
public static function executeUpdate($sql)

```

```

/**
 * Tranzakció indítása
 * @return void
 */
public static function beginTransaction()

/**
 * Tranzakció végrehajtása
 * @return void
 */
public static function commitTransaction()

/**
 * Tranzakció visszavonása
 * @return void
 */
public static function rollbackTransaction()

/**
 * Utolsó SQL utasítás által érintett sorok száma
 * @return integer
 */
public static function getUpdateCount()

/**
 * Utolsó létrehozott rekord azonosítója
 * @return integer
 */
public static function getLastInsertID()

/**
 * Levédi egy stringben a speciális karaktereket egy SQL lekérdezés számára
 * @param string $string
 * @return string
 */
public static function escape($string)

/**
 * Visszaadja az adatbázis kapcsolatot reprezentáló változót
 * @return resource
 */
public static function getResource()

/**
 * Singleton objektumot adja vissza
 * @return object
 */
public static function getInstance()
}

```

6.2.2 HTTP kommunikáció

A HTTP kommunikációt a Socket osztály valósítja meg. A külső adatbázisokhoz való interfész HTTP kérések küldésével és a kérésre adott válaszok segítségével történik. Az alábbi osztály egy olyan komponenst valósít meg, amely tetszőleges HTTP kérések kezelését valósítja meg.

```

Class Socket
{
/**
 * @var singleton $instance
 * @desc Singleton változó
 */
private static $instance;

/**

```



```

* @var resource $connection
* @desc Kapcsolat erőforrás
*/
private $connection = null;

/**
* @var string $connectionState
* @desc Kapcsolat állapota
*/
private $connectionState = DISCONNECTED;

/**
* @var int $defaultHost
* @desc Alapértelmezett IP amihez kapcsolódik
*/
private $defaultHost = "10.46.236.32";

/**
* @var int $defaultPort
* @desc Alapértelmezett port amin keresztül kapcsolódik
*/
private $defaultPort = 80;

/**
* @var float $defaultTimeout
* @desc Alapértelmezett időkorlát érték (mp)
*/
private $defaultTimeout = 10;

/**
* @var bool $persistentConnection
* @desc Meghatározza a kapcsolat típusát, true esetén folyamatos kapcsolat
*/
private $persistentConnection = false;

private $HTTP_Header;
private $HTTP_Body;

/**
* Konstruktor
* @return void
* @access private
*/
private function __construct()

/**
* Singleton minta. Minden hívónak ugyanazt azt a példányt adja vissza
* @return Socket
*/
public static function getInstance()

/**
* A megadott címhez a megfelelő porton próbál kapcsolatot létesíteni
* @return void
*/
public function connect($serverHost=false, $serverPort=false, $timeOut=false)

/**
* Lecsatlakozik a szerverről
* @return boolean True esetén sikeres, false esetén a kapcsolat már korábban
befejeződött
*/
public function disconnect()

```

```

/**
 * Parancs küldése a szervernek
 * @return string Szerver válasza
 */
public function sendCmd($command)

/**
 * Visszaadja a szerver válaszát (egysoros)
 * @return string Szerver válasza
 */
public function getResponse()

/**
 * Visszaadja a szerver többsoros válaszát
 * @return string Szerver válasza
 */
public function getMultilinedResponse()

/**
 * Egy sort olvas
 * @return string Szerver válasza
 */
public function readLine()

/**
 * Kapcsolat állapotának ellenőrzése
 * @return bool
 */
private function validateConnection()

/**
 * Kivételt dobása
 * @return void
 */
private function _throwError($errorMessage)

/**
 * Ha még volt élő kapcsolat, leválasztás
 */
public function __destruct()

private function decodeHeader()

private function decodeBody($endOfLine = "\r\n")

/**
 * Egy HTTP kérést hajt végre
 * @param string $url A kérés elérési útvonala
 * @return void
 */
public function doHTTPGet($url)

/**
 * A végrehajtott kérésre adott válasz fejlécét adja vissza
 * @return string
 */
public function getHTTPHeader()

/**
 * A végrehajtott kérésre adott válasz törzsét adja vissza
 * @return string
 */
public function getHTTPBody()
}

```

6.2.3 Naplózás

A naplózó komponens szintén szervizként van jelen a programban. Segítségével a program különböző állapotaiban a rendszernaplóba bejegyzések írhatóak, a futás közben az egyes részfeladatok végrehajtásáról az eredmények a képernyőn megjeleníthetők, a futás során keletkezett kivételek részletei elmenthetők. Továbbá lehetőséget biztosít az elmentett bejegyzések visszakeresésére.

```

Class Log
{
    /**
     * Singleton példány inicializálása
     * @return void
     */
    public static function initialize()

    /**
     * Bejegyzés mentése a rendszernaplóba
     * @param integer $type Az üzenet típusa LOG_INFO | LOG_ERROR | LOG_EXCEPTION
     * @param string $text A bejegyzése szövege
     * @param string $queryId A lekérdezés azonosítója
     * @return void
     */
    public static function addLog ($type, $text, $queryId='')

    /**
     * Üzenet megjelenítése a képernyőn
     * @param integer $type Az üzenet típusa LOG_INFO | LOG_ERROR | LOG_EXCEPTION
     * @param string $text A bejegyzése szövege
     */
    public static function screenLog ($type, $text)

    /**
     * Üzenetek lekérdezése a rendszernaplóból
     * @param integer $type Az üzenet típusa
     * @param string $from A kezdő dátum
     * @param string $to A vége dátum
     * @param string $queryId A lekérdezés azonosítója
     */
    public static function getLog ($type, $from, $to, $queryId='')
}

```

6.2.4 Felhasználó-azonosítás

A többfelhasználós működés támogatására a rendszer rendelkezik egy komponenssel, amely a felhasználók azonosítását és nyomon követését végzi. A program elindításakor a felhasználónak be kell írnia a felhasználónevét és jelszavát.

```

Class Authentication
{
    /**
     * visszaadja, hogy az adott munkamenetben a felhasználó bejelentkezett-e
     * @return boolean
     */
    public function loggedIn()
}

```

```

/**
 * Felhasználó beléptetése a megadott felhasználónévvel, jelszóval
 * @param string $username
 * @param string $password
 * @return boolean
 */
public function login ($username, $password)

/**
 * Felhasználó aktivitásának nyomonkövetése
 * @return void
 */
public function trackUser()

/**
 * Kijelentkezés
 * @return void
 */
public function logout()

/**
 * Hibakezelő, hiba esetén hívódik meg
 * @param integer $errorCode
 * @param string $errorDescription
 * @return void
 */
protected function error($errorCode, $errorDescription)

/**
 * Hibák lekérdezésére szolgáló eljárás
 * @return string
 */
public function getError()
}

```

6.2.5 NCBI Entrez kereső szolgáltatás

Az NCBI Entrezen történő keresés két részből épül fel. Az első lépés során a megadott kulcsszóval a megadott adatbázisban történik a keresés eredményhalmazának meghatározása. Az eredményhalmaz elemei a kapott találatok Entrez azonosítói. Második lépésben a egyes találatok lekérdezése történik. A találatok különböző formátumokban kérdezhetőek, például: FASTA, egyszerű szöveg, egyszerűsített XML. A programban megvalósított komponens az egyszerű XML változatot kérdezi le, amelyből a különböző annotációk kiolvashatók. Ezek tartalmazzák, hogy a találat melyik fajra, annak melyik génjére vonatkozik és tartalmazza a releváns szekvenciát is.

A kereső osztály:

```

class Bio_Entrez_eSearch
{
/**
 * Az Entrez eSearch szolgáltatásának elérési útvonala
 * @var string
 */
private static $eSearchURL =
    'http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?';

/**
 * A keresendő adatbázis neve (pl.: 'nuccore')
 * @var string
 */
private $database;

```

```

/**
 * A kereső kifejezés
 * @var string
 */
private $keyword;

/**
 * A keresés eredményeként kapott találatok száma
 * @var integer
 */
private $resultCount;

/**
 * A keresés eredményeként kapott találatok azonosítóit tartalmazó tömb
 * @var array
 */
private $results;

/**
 * @param string $keyword
 * @param string $database
 */
public function __construct($keyword, $database)

/**
 * A keresést végrehajtó metódus
 * @return boolean A keresés eredményeként kapott találatok száma, amennyiben
 *                 nagyobb, mint nulla, a visszatérési érték true, egyébként
 *                 false
 */
public function search()

/**
 * A keresés eredményeként kapott találatok számát visszaadó eljárás
 * @return integer
 */
public function getResultCount()

/**
 * A keresés eredményeként kapott találatok azonosítóit tartalmazó tömböt
 * visszaadó eljárás
 * @return array
 */
public function getResultIDs()
}

```

Az találat részletes adatait letöltő, reprezentáló osztály:

```

class Bio_Entrez_eFetch
{
/**
 * Az Entrez eSearch szolgáltatásának elérési útvonala
 * @var string
 */
private static $eFetchURL =
    'http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?';

/**
 * A lekérdezés formátuma
 * @var string
 */
private $retrieveMode      = 'xml';
private $retrieveType      = 'gb'; // GenBank

private $database;
private $id;

```

```

/**
 * @param string $database A találatot tartalmazó adatbázis neve
 *                        (pl.: 'nuccore')
 * @param integer $id A találat Entrez azonosítója
 */
public function __construct($database, $id)

/**
 * A lekérdezés végrehajtása
 * @return boolean
 */
public function fetch()

/**
 * A találatban szereplő fajnév
 * @return string
 */
public function getSpeciesName()

/**
 * A találatban szereplő kromoszóma
 * @return integer
 */
public function getChromosome()

/**
 * A találatban szereplő gén azonosító
 * @return string
 */
public function getGeneId()

/**
 * A találatban szereplő külső adatbázis (pl.: HGNC, MGI) neve
 * @return string
 */
public function getExternalDatabaseName()

/**
 * A találatban szereplő gén külső adatbázisban található azonosítója
 * @return string
 */
public function getExternalDatabaseId()

/**
 * A találatban szereplő szekvencia
 * @return string
 */
public function getSequence()
}

```

6.2.6 BLAST szolgáltatás

A munkafolyamat során lehetőség van egyes lépésekben talált adatok további keresésére. Ez az NCBI online BLAST szolgáltatás használatának segítségével válik lehetővé. Az egyes explicit gén találatokból a szekvencia kinyerhető, és ez felhasználható további keresésekre más fajok genomjában. Az egyes szignifikáns találatok további lehetőségeket a végeredmény szempontjából.

A BLAST keresés időigényes folyamat lehet, ezért a paraméterek megfelelő megválasztásával kell a szolgáltatást igénybe venni. Az osztály legfontosabb metódusa a `search()`, amiben egy végtelen ciklus köré szerveződve hajtódik végre a keresés, amely segítségével a BLAST keresés eredményére vár az algoritmus.

```

class Bio_NCBI_Blast
{
/**
 * Az NCBI Blast szolgáltatásának elérési útvonala
 * @var string
 */
private static $blastURL = 'http://www.ncbi.nlm.nih.gov/blast/Blast.cgi';

private $program;

private $url;

private $return;

private $results;

/**
 * @param string $sequence A keresendő szekvencia
 * @param string $program A BLAST program (megablast, blastn, blastp,
 * rpsblast, blastx, tblastn, tblastx)
 * @param string $database A keresendő adatbázis (pl.: 'nucore')
 */
public function __construct($sequence, $program, $database)

/**
 * A megadott szekvenciát FASTA formátumba konvertálja
 * @param string $sequence
 * @return string
 */
private function sequence2Fasta($sequence)

/**
 * A kereséshez paramétert rendel hozzá (például: Elvárt érték)
 * @param string $key
 * @param string $value
 * @return void
 */
public function addParameter($key, $value)

/**
 * A kiválasztott program és paraméterek alapján összeállítja a lekérdezés
 * URL-jét
 * @return string
 */
private function createSearchURL()

/**
 * Végrehajtja a keresést
 * @return void
 */
public function search()

/**
 * Tájékoztat a keresés végeredményéről
 * 0 - sikeres keresés
 * 1 - érvénytelen paraméterek
 * 2 - nincs szignifikáns találat
 * 3 - a keresés azonosító (RID) lejárt
 * 4 - a keresés megghiúsult
 * 5 - ismeretlen hiba
 * @return integer
 */
public function getReturnCode()

/**
 * A keresés eredményét adja vissza
 * @return string
 */
public function getResults()
}

```

6.2.7 BioMart szolgáltatás

Az előzőekben bemutatott BioMart Martservice szolgáltatás szintén HTTP kérésen keresztül érhető el. A rendszer egy meghatározott formátumú XML-t vár bemenetként, amire egy táblázatot ad válaszul. A táblázat egyes elemei tabulátor jellel (”\t”) vannak elválasztva egymástól. A dolgozat keretein belül megvalósított komponens képes a lekérdezéshez szükséges XML-t összeállítani, illetve a kapott választ értelmezni és azt asszociatív tömb formájában visszaadni.

```
class Bio_BioMartQuery
{
/**
 * A központi BioMart szerver Martservice elérési útvonala
 * @var string
 */
private static $martServiceURL =
    'http://www.biomart.org/biomart/martservice?query=';
private $query;
private $queryXML;

/**
 * Beállítja az adathalmazt
 * @param string $name
 * @return void
 */
public function setDataset($name)

/**
 * Hozzáad egy filtert a lekérdezéshez
 * @param string $name
 * @param string $value
 * @return void
 */
public function addFilter($name, $value)

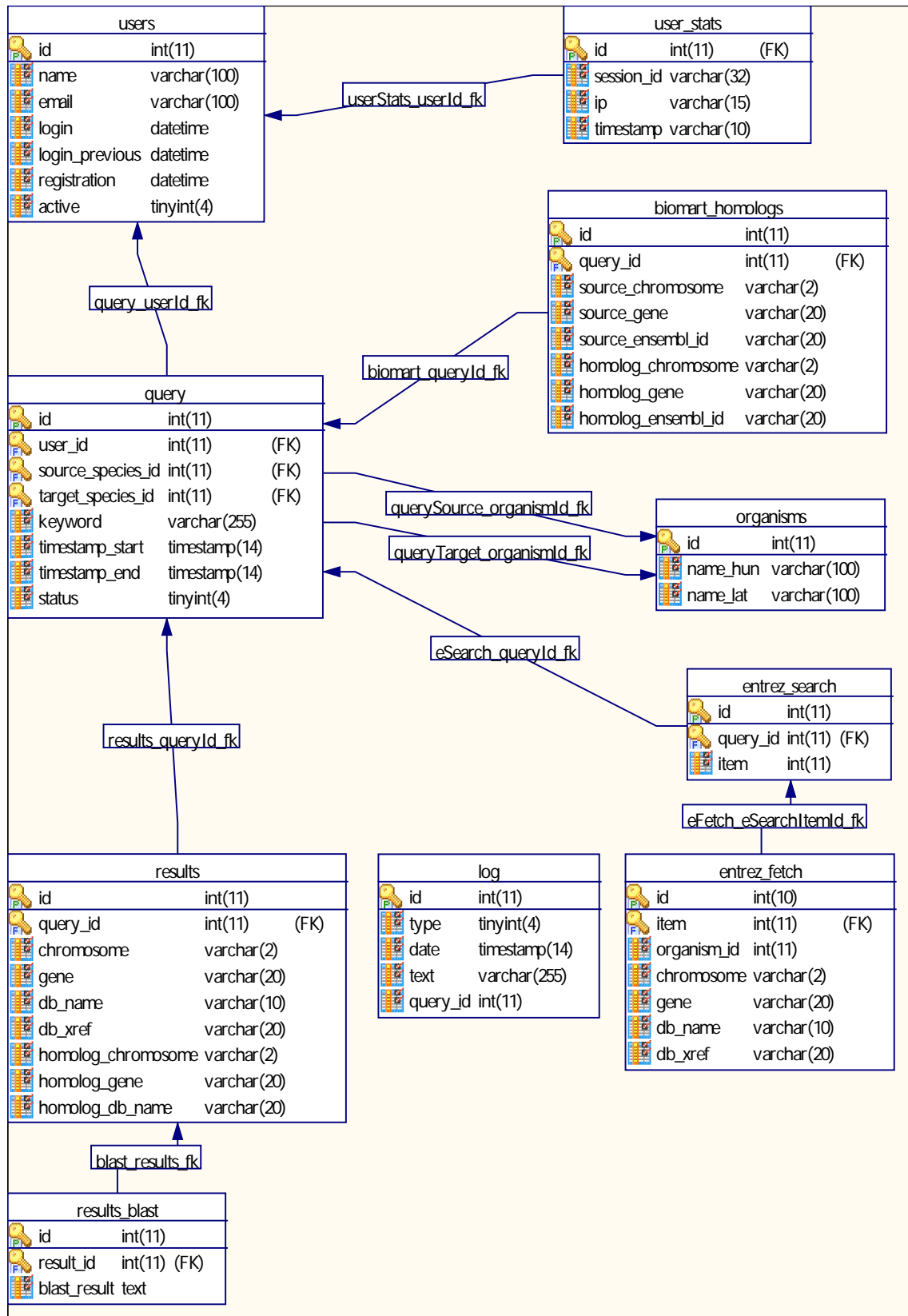
/**
 * Hozzáad egy attribútumot a lekérdezéshez
 * @param string $name
 * @return void
 */
public function addAttribute($name)

/**
 * Összeállítja $this->queryXML-be a lekérdezés XML-jét
 * @return void
 */
protected function buildQuery()

/**
 * A HTTP kérésre adott tabulált táblázatot asszociatív táblázatba alakítja át
 * @return string
 */
protected function parseResults()

/**
 * Végrehajtja a lekérdezést
 * @return boolean
 */
public function execute()
}
```


6.3 Adatstruktúra



19. ábra. A program adattáblái és a közöttük lévő relációk

6.4 Felhasználói felület

A szoftver grafikus felhasználói felülettel rendelkezik, amely különböző régiókra osztható. A fejlécben a felhasználóval kapcsolatos információk láthatóak, valamint itt található a főmenü. A főmenüben a rendszer különböző szolgáltatásai érhetőek el (új keresés indítása, eredmények megtekintése, munkafolyamat szerkesztés). Az oldal tartalmi részében jelennek meg a kiválasztott menüpontnak megfelelő tartalmak. Legalul a lábléc található. (20. ábra)



20. ábra. A felhasználói felület egyes régiói

7. A megvalósított rendszer

21. ábra. A megvalósított rendszer grafikus keresőfelülete

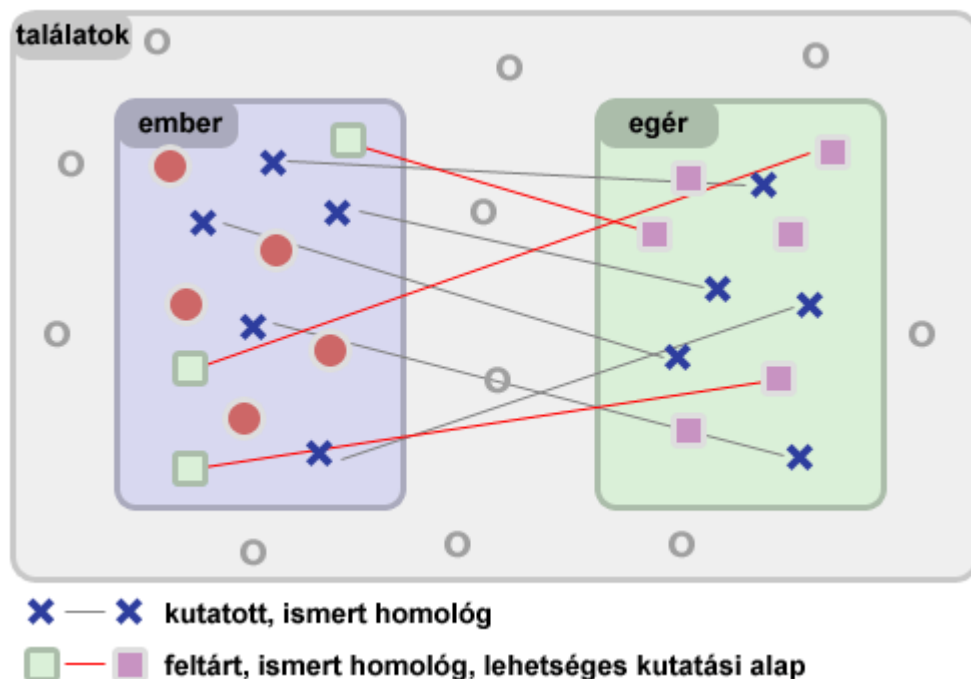
7.1 Implementálási környezet

A program PHP szkriptnyelven íródott, az eredményeket MySQL 5.0-ban tárolja. Választásom azért esett erre a webes szkriptnyelvre, mert magas szintű nyelv, amelynek sok olyan beépített szolgáltatása van (például XML feldolgozás), amely a rendszer megvalósítását könnyebbé teszi. A külső modulként beépülő PHP modult Apache webservert futtatja, továbbá ez szolgálja ki az egyes események által kiváltott kéréseket. Ezen szoftverek különböző operációs rendszerekre telepíthetőek, így a program könnyen hordozhatóvá válik. Jelen esetben a fejlesztést és tesztelést Windows-os környezetben végeztem, míg a futtatást Linuxon.

7.2 Kutatási cél

A szoftver működését szemléltető példa egy biológiai vonatkozású adatbányászat folyamatot valósít meg, amelynek során az asztma betegséggel kapcsolatos fehérjék emberbeli kódoló régióinak feltérképezése, valamint az egér modell állat genom szintű a betegséggel logikailag összefüggő nukleotid kódoló régióinak feltérképezése és a homológiai szabályok alapján a gén megfeleltetések grafikus ábrázolása valósult meg.

A kutatás konkrét célja olyan kódoló régiók feltárása, amelyek az eddigi kutatások során az egyik organizmusban egyértelműen a megadott betegséggel kapcsolatban álltak, viszont a másik organizmusban az adott régióval kapcsolatban semmilyen betegséggel kapcsolatos dolgot nem fedeztek még fel. Ennek biológiai hátterét a törzsfajlódás elve, az egyes közös ősök létezése biztosítja. Az egyes találatok, azaz a találatokban található gének, nukleinsav szekvenciák jelen értelmezés szerint három csoportba oszthatók.



22. ábra. A találatok grafikus értelmezése

Az első csoportba azok a találatok tartoznak, amelyekben az adott betegséggel kapcsolatban az összefüggés a kiindulási és cél fajban is ismertek. Ezek a jelen kutatás szempontjából nem tekinthetők szignifikánsnak, mert a betegség és az adott gén kapcsolata már kutatott, az összefüggés ismert. Ezt a 22. ábrán a kék keresztek jelölik.

Második csoportba azok a találatok sorolhatóak, amelyekben, a cél organizmusban (modellállatban) a betegséggel összefüggésbe hoztak géneket, de ezek a kiindulási organizmusban nem találhatók meg. Ennek a jelenségnek kézenfekvő biológiai magyarázatát a fajok közötti genetikai eltérése adja. Ezek a találatok szintén nem tekinthetők hasznosnak, mert a kitűzött cél szempontjából nem hasznosíthatóak. Ezeket a találatokat a fentebb látható ábrán olyan lila négyzetek szimbolizálják, amelyekhez nem kapcsolódik piros egyenes.

A harmadik csoportot a találatoknak az a halmaza alkotja, amelynek elemei a keresett kifejezéssel kapcsolatban a cél organizmusban összefüggésbe hozhatók, ugyanakkor a találatban szereplő génnek vagy nukleinsav szekvenciának nincs kutatott eredménye (találata az adatbázisban) a kiindulási organizmussal kapcsolatban. Ezek a találatok tekinthetők hasznosnak, mert az adatok megfelelő értelmezéséből olyan új információt hoznak létre, amely a jövőben újabb tudás megismerésére adhatnak lehetőséget. Ezeket a találatokat az ábrán azok a négyzetek jelölik, amelyek a másik élőlény halmazában található négyzetekkel piros vonallakkal vannak összekötve.

A halmazos értelmezést formálisan megadva:

$H := \{ \text{emberrel kapcsolatos találatok, modell állat homológ megfelelői} \}$

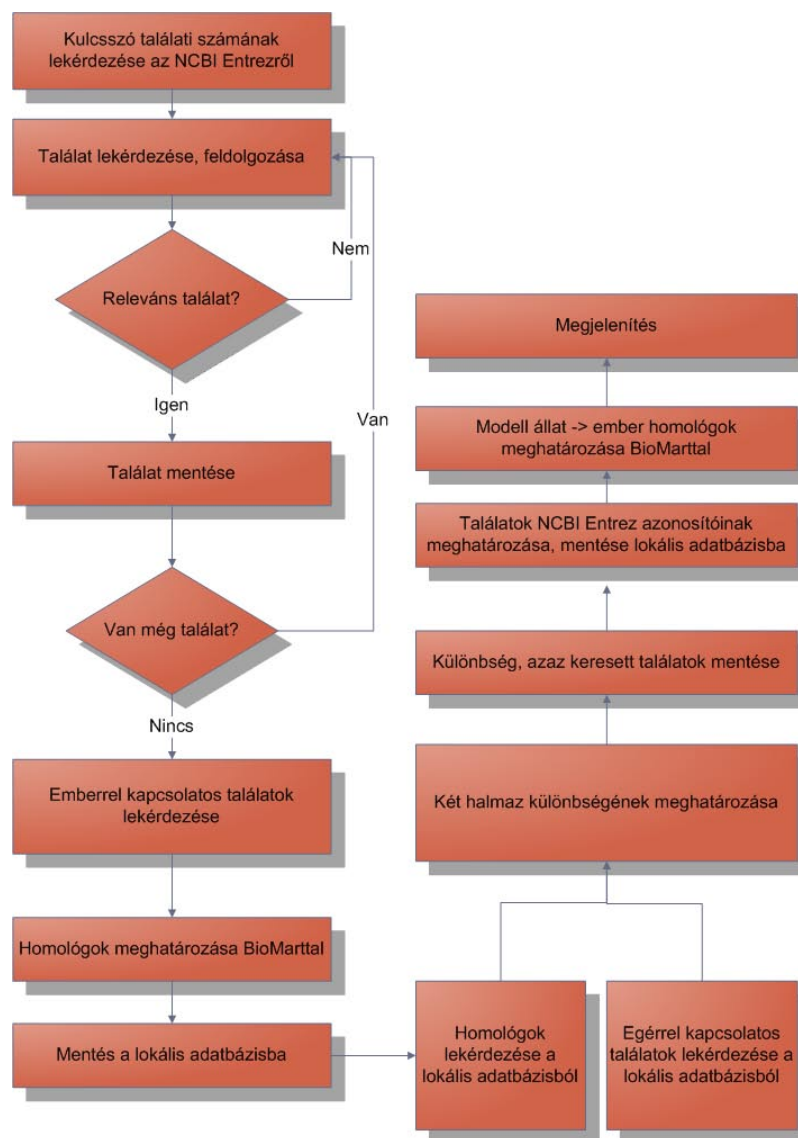
$E := \{ \text{egérrel kapcsolatos találatok} \}$

$T := E \setminus H$

Ahol T jelöli a hasznos találatokat, amelyek az adatbányászat során a keresett információ alapját alkotják. Az előző lépésben a gének összehasonlíthatóvá váltak, így a halmazművelet könnyen elvégezhető, előáll a találati halmaz.

7.3 A munkafolyamat bemutatása egy konkrét példa segítségével

A lekérdezés folyamatát a 19. ábra szemlélteti.



23. ábra. A bemutatott példa munkafolyamata

A keresés folyamata során a program más bioinformatikai adatbázisokkal lép kapcsolatba, illetve online elérhető szoftverekkel, amelyek önmaguk is több adatbázist integrálnak illetve biztosítanak átjárhatóságot az egyes rendszerek között. Az alábbiakban egy konkrét keresés folyamatát ismertetem szekvenciálisan. A kiindulási faj az ember, a kutatott modellállat pedig az egér, a keresett kifejezés az „*asthma*”. A keresés BLAST-olási opció nélkül történik. A keresés első lépéseként egy bejegyzést hoz létre a program a *query* táblában, amely a fajok azonosítóit illetve a keresés kulcsszavát tartalmazza.

7.3.1 Gén azonosítók lekérdezése

A munkafolyamat első lépésének keresési terét a bioinformatikai adatbázisok egyik központjának számító NCBI által kifejlesztett Entrez alkotja. Az Entrez számos adatbázist tartalmaz, a szoftver első lépésben a nukleotid adatbázisban keres. Ebben olyan bejegyzések találhatók, amelyek konkrét tudományos publikációk, amelyek bizonyos DNS szekvenciához is köthetők.

Az Entrezhez egy EUtils nevű API érhető el, amely HTTP kapcsolaton és megadott specifikáció szerint képes adatokat fogadni. A keresés két lépcsőben történik. Első lépésben az ESearch nevű rendszernek adja át a szoftver a kereső kifejezést. A visszaadott válasz egy XML fájl, amelyet a program feldolgoz, és kiemeli, hogy az adott kereső kifejezéshez hány találat érhető el az NCBI Entrez adatbázisában. A lekérdezett azonosítókat az *entrez_search* táblában tárolja a rendszer.

7.3.2 Azonosítók adatainak lekérdezése, feldolgozása

Rendelkezésünkre állnak a találatok azonosítói, következő lépésben a találatokat kell feldolgoznunk. Ezt az EFetch nevű szolgáltatással valósíthatjuk meg. Az azonosítókat átadva az eredményeket különböző formátumban kérdezhetjük le. Az általunk megvalósított rendszer a GenBank formátumhoz hasonló formátumban kérdezi le a találatokat, de XML-ben. Az XML feldolgozása során a számunkra releváns információkat nyerjük ki, és csak akkor tudjuk elmenteni a találatot, ha az összes olyan információt megtaláljuk a bejegyzésben, amelyre nekünk szükségünk van. Sok olyan publikált találat van, amely a keresés szempontjából nem eléggé specifikus, nem találhatók meg benne a számunkra szükséges információk. A jelenlegi változatban a találatból a bejegyzéshez kapcsolódó élőlény fajnevét, a kutatás szempontjából érdekes kromoszóma és gén nevét mentjük el. A gének elnevezésére és hivatkozására több változat van, köszönhetően a sok bioinformatikai adatbázisnak. Amennyiben elérhető, akkor a program több annotációt is képes értelmezni és azokat elmenteni az *entrez_fetch* adattáblában.

7.3.3 Ember → egér homológok lekérdezése

Ahhoz, hogy meg tudjuk határozni, hogy melyek azok a találatok, amelyek egy későbbi esetleges kutatás alapjait képezhetik a modell állaton, tudnunk kell, hogy a betegséggel kapcsolatos az emberben ismert géneknek az egérben hol van a megfelelője, homológja. Ugyanis, ha tudjuk ezeket a homológokat, akkor ezeket kivonhatjuk a modell állattal kapcsolatos találatokból és így olyan találatok maradnak, amelyek egy későbbi kutatás alapját képezhetik.

Ahhoz, hogy az emberben található gének modellállatbeli megfelelőjét meghatározzuk, nem szükséges szekvencia illesztő algoritmusokat vagy más homológiát meghatározó eszközt igénybe vennünk, ezek az információk már rendelkezésre állnak, és elérhetőek nyilvános szoftverek segítségével. Ilyen alkalmazás például a BioMart. Meg kell határozni, hogy milyen élőlény genomjából indulunk ki (*Dataset*), milyen információkkal rendelkezünk (*Filters*), például gének, illetve milyen tudást szeretnénk megkapni (*Attributes*).

A webes felületen kívül a szoftverhez különböző API-k érhetőek el: BioPerl, Java és Webservice interfész. Mivel a mi programunk PHP-t használ, ezért az általános Webservice API-t választottuk. Ez egy XML lekérdezés, amelyben a fentebb említett információkat kell átadnunk és az alkalmazás visszaad egy táblázatot, amelynek oszlopait a kiválasztott attribútumok alkotják.

A mi esetünkben a releváns emberekkel kapcsolatos találatok génjeinek hivatalos (*HGNC*) azonosítóját adtuk át a BioMart-nak és az egér beli homológ kromoszómát és a homológ gén hivatalos nevét (*MGI*) kérdeztük le. Az eredményeket *biomart_homologs* táblában tárolja a program.

7.3.4 Találatok Entrez azonosítóinak meghatározása

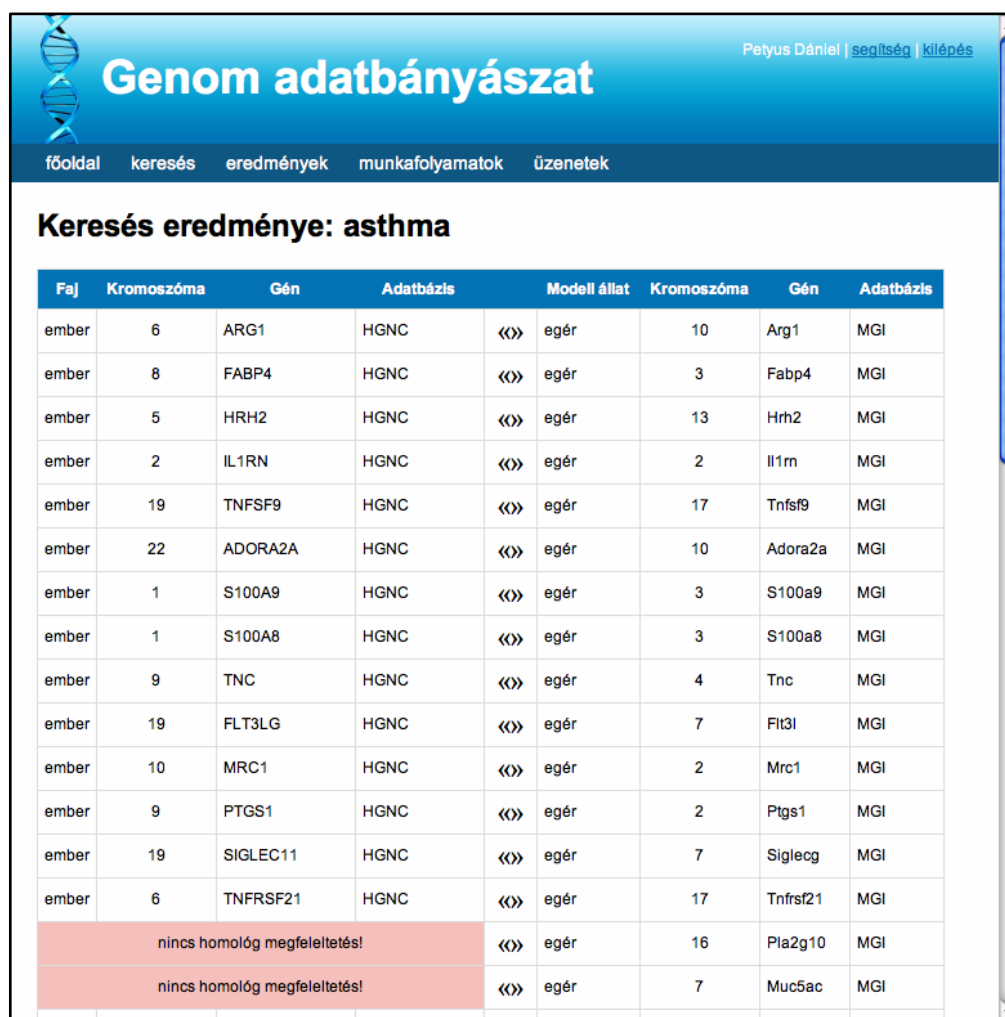
Ez a lépés egy szükségszerűségből közbeiktatott fázisa a munkafolyamatnak, mert a BioMarttal meghatározott egér azonosítókat a BioMart rendszer fordított irányban nem fogadja el, ezért egy köztes lépésként a szoftver ismét az NCBI Entrez adatbázishoz fordul és olyan azonosítókat keres, amelyet visszafelé elfogad a BioMart.

7.3.5 Egér → ember homológok lekérdezése

A végeredményhez szükséges, hogy meghatározzuk a találatok emberbeli homológját. Ugyanis, ha az emberben nem található a modellálltban meghatározott gén homológja, annak különböző okai lehetnek. Lehetséges, hogy még nem meghatározott az a gén, de lehet, hogy a törzsfjlődés során történt változások miatt nem is létezik az emberben. Utóbbi esetben természetesen emberrel kapcsolatos kutatás alapját nem képezheti. A már ismerttetett folyamatot ismét végigjárva, más attribútumokkal a kért információk lekérdezhetők. A szűrt információk rendezett formában a *results* táblába kerülnek. Amennyiben a keresés során a BLAST lehetőséget is választjuk, akkor az értékes találatok szekvenciáit a rendszer BLAST-olja, és az eredményes BLAST keresések riportjai a *results_blast* táblában tárolja.

7.3.6 Eredmények megjelenítése

A létrejött eredményhalmaz tartalmazza, az esetleges kutatás alapját képező géneket a modell állatban, illetve annak megfelelőjét az emberben. A gének elhelyezkedése az egyes kromoszómákon is ismert, így az eredmények táblázatosan, vagy grafikusan összerendelve is megjeleníthetővé válnak. (24. ábra)



The screenshot shows a web application titled 'Genom adatbányászat' with a navigation bar containing 'főoldal', 'keresés', 'eredmények', 'munkafolyamatok', and 'üzenetek'. The main content area is titled 'Keresés eredménye: asthma' and displays a table of search results comparing human and mouse genes.

Faj	Kromoszóma	Gén	Adatbázis		Modell állat	Kromoszóma	Gén	Adatbázis
ember	6	ARG1	HGNC	«»	egér	10	Arg1	MGI
ember	8	FABP4	HGNC	«»	egér	3	Fabp4	MGI
ember	5	HRH2	HGNC	«»	egér	13	Hrh2	MGI
ember	2	IL1RN	HGNC	«»	egér	2	Il1rn	MGI
ember	19	TNFSF9	HGNC	«»	egér	17	Tnfsf9	MGI
ember	22	ADORA2A	HGNC	«»	egér	10	Adora2a	MGI
ember	1	S100A9	HGNC	«»	egér	3	S100a9	MGI
ember	1	S100A8	HGNC	«»	egér	3	S100a8	MGI
ember	9	TNC	HGNC	«»	egér	4	Tnc	MGI
ember	19	FLT3LG	HGNC	«»	egér	7	Flt3l	MGI
ember	10	MRC1	HGNC	«»	egér	2	Mrc1	MGI
ember	9	PTGS1	HGNC	«»	egér	2	Ptgs1	MGI
ember	19	SIGLEC11	HGNC	«»	egér	7	Siglecg	MGI
ember	6	TNFRSF21	HGNC	«»	egér	17	Tnfrsf21	MGI
nincs homológ megfeleltetés!				«»	egér	16	Pla2g10	MGI
nincs homológ megfeleltetés!				«»	egér	7	Muc5ac	MGI

24. ábra. A munkafolyamat találatainak megjelenítése

7.4 Módosítások az eredeti elképzeléshez képest

Az eredeti koncepciótól a megvalósított rendszer alapvetően két ponton tér el. Elképzelésünk szerint az NCBI Entrezon meghatározott találatokat a visszkapott szekvencia felhasználásával BLAST segítségével lokalizáljuk az egyes kromoszómákon. Azonban kiterjesztett keresés segítségével az egyes találatokból a gének és a kromoszómák közvetlenül lekérdezhetőek.

A második eltérés ellentétben az elsővel nem könnyebbséget jelent, hanem új, megoldandó feladatot iktat közbe. A probléma akkor jelentkezik, amikor az egér →

ember homológokat próbáljuk meghatározni BioMart segítségével. A rendszer bár felkínálja, hogy elfogadja a hivatalos nevezék és az MGI szerinti azonosítót, azonban így nem működik a szoftver. Ezért kellett egy új lépés a folyamatba beiktatnunk, amely során a hivatalos nevezék azonosítóhoz meghatározzuk az Entrez azonosítót, amellyel már működik „visszafelé” a BioMart.

7.5 Fejlesztési környezet

A fejlesztői gép hardware konfigurációja:

- Dell Inspiron 6400
- Intel Core 2 Duo T5600-as 1.83GHz
- 2 GB DDR2 SDRAMM

A program tesztelése, fejlesztése során használt szoftverkörnyezet:

- Windows Vista
- Apache 2.2 webservert
- PHP 5.2.0 interpreter
- MySQL 5.0.45 adatbázis kezelő rendszer

Hálózati kapcsolat:

- 10 Mbit/s letöltési, 1 Mbit/s feltöltési sebesség

7.6 Futtatási környezet

A futtatási környezet hardware konfigurációja

- Fujitsu-Siemens workstation
- Intel Pentium D 2.8 GHz
- 1 GB RAM

A program futtatása során használt szoftverkörnyezet:

- Fedore 5 Linux, 2.6-os kernel
- Apache 2.2
- PHP 5.1.2
- MySQL 5.0

Hálózati kapcsolat:

- gerinchálózati kapcsolat

7.7 Futási idők

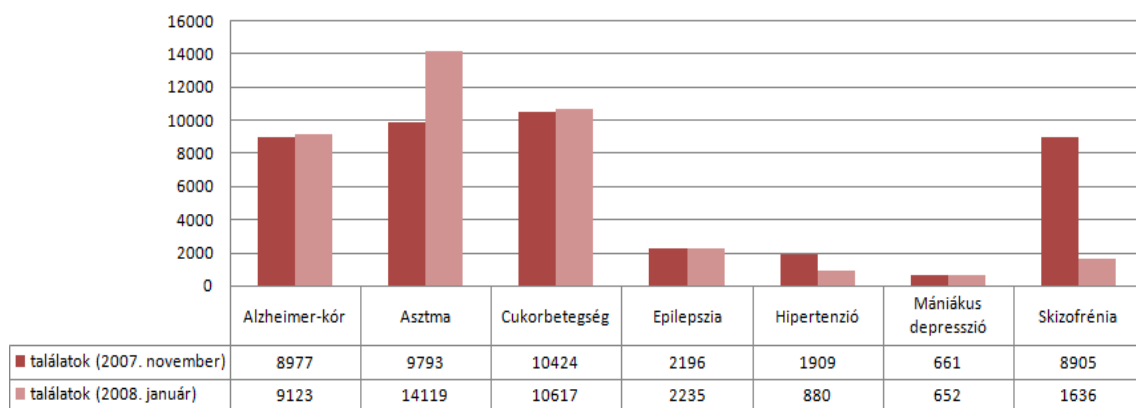
Fejlesztési környezet	ó:perc:mp
1. azonosítók lekérdezése	0:00:12
2. rekordok lekérdezése, feldolgozása	1:01:14
3. Ember → egér homológok lekérdezése	0:00:05
4. Találatok meghatározása	0:00:03
5. Találatok Entrez azonosítóinak meghatározása	0:04:32
6. Egér → ember homológok lekérdezése	0:00:05
Összesen:	1:06:11

Futtatási környezet	ó:perc:mp
1. azonosítók lekérdezése	0:00:07
2. rekordok lekérdezése, feldolgozása	1:04:41
3. Ember → egér homológok lekérdezése	0:00:03
4. Találatok meghatározása	0:00:03
5. Találatok Entrez azonosítóinak meghatározása	0:01:59
6. Egér → ember homológok lekérdezése	0:00:01
Összesen:	1:06:54

Jól látható, hogy a legtöbb időt a második lépés: a rekordok lekérdezése, feldolgozása jelenti. Ez annak tudható be, hogy a közel tízezer rekord lekérdezése és feldolgozása az NCBI szerverről igen időigényes feladat. A számítógépek számítási kapacitását nem veszi túlságosan igénybe egy ilyen rekord feldolgozása, ezért a rendszer ezen lépése kisebb kapacitású számítógépekre eloszthatóvá lenne tehető.

7.8 Kutatási eredmények

Az eredmények bemutatása előtt egy összefoglaló táblázattal szeretném szemléltetni az adatbázisok növekedését. (25. ábra) Az első mérést ezelőtt három hónappal végeztem, azóta az egyes kulcsszavakra adott találatok száma bizonyos kifejezésekre növekedett. A leglátványosabb növekedés az *asthma* kulcsszóra adott válaszok számában látható, ugyanakkor előfordult jelentős csökkenés is, például a *schizophrenia* esetében. Az adatok változása bizonyítja, hogy az adatbázis tartalmát gondozzák, új tartalmak kerülnek fel, a nem megfelelő találatokat eltávolítják a rendszerből. Ezért a dolgozat által végrehajtott keresést is célszerű megismételni a jövőben az egyes új potenciális találatok reményében.



25. ábra. A keresőszavakra adott találatok száma az Entrez nukleotid adatbázisban

Az alábbi táblázatok az előzőekben bemutatott konkrét példa eredményeit tartalmazzák. A végleges eredmények bemutatása előtt az alábbi táblázatban az egyes fázisokban tapasztalt találatok számával kapcsolatban szeretnék magyarázatot adni. A táblázat első felében az *asthma* kulcsszóra adott találatok száma látható, az Entrez *CoreNucleotide* adatbázisában. Noha majdnem tízezer találatról van szó, ezeknek csak nagyon kis része (1,73%-a) tartalmazza azokat a szükséges információkat, amelyek a munkafolyamat végrehajtása során szükségesek. A többi adat valószínűleg ellenőrizetlenül, automatikus beküldés útján került be a rendszerbe, ezeket a találatokat a munkafolyamat során nem lehet hasznosítani.

A 169 kiértékelhető találatból 113 találat vonatkozik az emberre, és 56 az egérre. A második fázisban, a homológiák meghatározása során a 113 emberrel kapcsolatos találatra a BioMart 122 egérbeli homológ gént adott válaszul. Ezt összevetve az első keresésből származó 56 találattal, a két találathalmaz különbsége 42 egérben található gén.

Asthma találatok az NCBI-on		
Összesen:	9793	100%
Hasznos találatok:	169	1,73%
Nem értékelhető:	9624	98,27%
Kiértékelhető:	169	100%
Ember:	113	66,86%
Egér:	56	33,14%
Ember → egér homológok BioMart szolgáltatás által		
Ember	113	100%
Egér	122	107,96%
Egér homológok	122	
Egér gének	56	
"Hasznos" egér gének	42	

A végleges eredményeket, az alábbi táblázat tartalmazza. A táblázatban olyan rekordok találhatóak, amelyeket a kutatási célokban megfogalmazott besorolás szerint a második és harmadik csoportba sorolhatóak. Egy találatot egy sor reprezentál a

táblázatban, azoknál a találatoknál, amelyek második csoportba tartoznak, „nincs homológ megfeleltetés” látható a kiindulási organizmus oszlopaiban, míg a táblázat többi eleme a harmadik kategóriába tartozik, tehát pozitív találat.

Faj	Kromosz.	Gén	Adatb.		Modellá.	Kromosz.	Gén	Adatb.
ember	6	ARG1	HGNC	«»	egér	10	Arg1	MGI
ember	8	FABP4	HGNC	«»	egér	3	Fabp4	MGI
ember	5	HRH2	HGNC	«»	egér	13	Hrh2	MGI
ember	2	IL1RN	HGNC	«»	egér	2	Il1rn	MGI
ember	19	TNFSF9	HGNC	«»	egér	17	Tnfsf9	MGI
ember	22	ADORA2A	HGNC	«»	egér	10	Adora2a	MGI
ember	1	S100A9	HGNC	«»	egér	3	S100a9	MGI
ember	1	S100A8	HGNC	«»	egér	3	S100a8	MGI
ember	9	TNC	HGNC	«»	egér	4	Tnc	MGI
ember	19	FLT3LG	HGNC	«»	egér	7	Flt3l	MGI
ember	10	MRC1	HGNC	«»	egér	2	Mrc1	MGI
ember	9	PTGS1	HGNC	«»	egér	2	Ptgs1	MGI
ember	19	SIGLEC11	HGNC	«»	egér	7	Siglecg	MGI
ember	6	TNFRSF21	HGNC	«»	egér	17	Tnfrsf21	MGI
nincs homológ megfeleltetés!				«»	egér	16	Pla2g10	MGI
nincs homológ megfeleltetés!				«»	egér	7	Muc5ac	MGI
ember	22	IL17RA	HGNC	«»	egér	6	Il17ra	MGI
ember	21	TFF1	HGNC	«»	egér	17	Tff1	MGI
ember	21	ITGB2	HGNC	«»	egér	10	Itgb2	MGI
ember	17	EPX	HGNC	«»	egér	11	Epx	MGI
ember	3	CCR8	HGNC	«»	egér	9	Ccr8	MGI
ember	1	IL19	HGNC	«»	egér	1	Il19	MGI
ember	15	SPRED1	HGNC	«»	egér	2	Spred1	MGI
ember	1	TNFRSF18	HGNC	«»	egér	4	Tnfrsf18	MGI
ember	18	HRH4	HGNC	«»	egér	18	Hrh4	MGI
nincs homológ megfeleltetés!				«»	egér	7	Siglec5	MGI
ember	16	IL21R	HGNC	«»	egér	7	Il21r	MGI
ember	4	NMU	HGNC	«»	egér	5	Nmu	MGI
ember	19	EBI3	HGNC	«»	egér	17	Ebi3	MGI
ember	11	PRG2	HGNC	«»	egér	2	Prg2	MGI

Faj	Kromosz.	Gén	Adatb.		Modellá.	Kromosz.	Gén	Adatb.
ember	11	MUC5B	HGNC	«»	egér	7	Muc5b	MGI
ember	X	NP_001020436.1	RefSeq_peptide	«»	egér	5	Tpte2	MGI
ember	1	NP_060887.1	RefSeq_peptide	«»	egér	1	Sacy	MGI
ember	13	F10	HGNC	«»	egér	8	F10	MGI
ember	1	CYP4A22	HGNC	«»	egér	4	Cyp4a12b	MGI
ember	5	CAMLG	HGNC	«»	egér	13	Caml	MGI
nincs homológ megfeleltetés!				«»	egér	1	Serpinb3a	MGI
ember	5	TNIP1	HGNC	«»	egér	11	Tnip1	MGI
nincs homológ megfeleltetés!				«»	egér	7	EG668526	MGI
nincs homológ megfeleltetés!				«»	egér	7	EG624439	MGI
nincs homológ megfeleltetés!				«»	egér	7	EG624584	MGI
nincs homológ megfeleltetés!				«»	egér	1	mIL-17Ftv	MGI

8. További perspektívák

A meglévő szoftver továbbfejlesztésére több lehetséges mód kínálkozik. Az egyes visszajelzésektől függően a rendszer fejlesztése különböző irányokba mozdítható el.

Elképzelhető, hogy a felhasználóknak a munkafolyamatok szerkesztésére lenne nagyobb szükségük, ebben a esetben egy *workflow* szerkesztő felület létrehozása lenne a cél. A munkafolyamatok komplexitásának növekedése, a program használatának intenzitásának növekedése a számításigénye növekedését vonja magával, így a rendszert szükséges lenne skálázhatóvá tenni. Ez olyan architektúrális megoldást jelent, amely biztosítaná, hogy a szoftver egyszerre több szerveren fusson, és az egyes folyamatok külön gépekre kerüljenek. Ez a fajta terhelés megosztás más módokon is kivitelezhető, például *grid* rendszeren történő futtatással is, amely a folyamat szintjén is lehetővé tenné a párhuzamosítást. Nem utolsósorban a rendszer a kollaboráció irányába is fejleszthető, amely az egyes felhasználók közötti kommunikáció, tudásmegosztás magasabb szintű támogatásával tenné hatékonyabbá a munkát.

9. Összefoglalás

A dolgozat témája egy ma Magyarországon kevésbé oktatott és kutatott területhez kapcsolódik, a bioinformatikához. A bevezetőben megfogalmazott biológiai fogalmak rövid összefoglalása a dolgozatban tárgyalt problémához szükséges alapok elhelyezése mellett a téma iránt érdeklődő informatikusok számára egy olyan betekintést nyújthat, amely az érdeklődésüket felkeltheti, és ez által újabb művelőket toborozhat a bioinformatikát kedvelők táborába. Ezen szakterület bemutatása során igyekeztem megfogalmazni a tudományok és a határtudományok mezsgyéjén a bioinformatika létjogosultságát, összevetve egy más fajta – számomra is elfogadhatóbb – szemlélettel. A program használt szolgáltatások alapjául szolgáló algoritmusok áttekintését adtam meg ezután. Itt elsősorban olyan szekvencia illesztő algoritmusok leírását igyekeztem vázlatosan – a megértéshez szükséges szinten – összefoglalni, amelyet a program által használt szoftverek a szekvenciák keresése során használnak.

Ezek után a kitűzött probléma megoldásához az egyes szükséges eszközöket mutattam be, elemeztem őket használhatóságuk szempontjából. Ezek a leírások azoknak az online elérhető, programozási interfésszel rendelkező szolgáltatásoknak a rövid bevezetője, amelyek a kitűzött cél megvalósításához a munkafolyamatban felhasználtam. Sajnálatos módon a dolgozatban használt szolgáltatás orientált keresési megoldásokkal kapcsolatban jelenleg hozzáférhető irodalom, jegyzet, közösségi tudás nincs publikált formában magyar nyelven, ezért is láttam fontosnak az ismeretek ilyen szintű rendezettségének megalapozását. Az itt bemutatott eljárások a bevezető szintű laborgyakorlatokra épülve, de egy magasabb szinten próbálják a kitűzött cél elérését szolgálani.

A mérnöki szemléletet szem előtt tartva, a munka során a tervezési fázist a probléma megoldásának tervezési terének megadása előzi meg. Ebben a részben különböző lehetséges megoldási módokat mutatok be, amelyek a dolgozat által kitűzött célokat kielégítik. Ezt egy indokolt döntés meghozása követi, ami kijelöli a megvalósítás módját. A megvalósított szoftver egy PHP nyelven íródott program, amely különböző, eddig nem létező komponenseket tartalmaz, amelyek segítségével az előzőekben tárgyalt bioinformatikai szoftverek használhatóak, az eredmények MySQL adatbázisban kerülnek tárolásra. A tervezési fázisban a rendszerben fellelhető komponensek az objektum-orientált programozási szemlélet szerint kerültek megalkotásra. Az adatstruktúra kialakítása során relációs adatbázisok minden lehetőségét kiaknázva igyekeztem egy olyan átlátható, letisztult adatbázis szerkezetet tervezni, amely megfelelően optimális és könnyen továbbfejleszthető.

A megvalósított program az eredeti koncepció által kijelölt kritériumoknak megfelel, egy olyan adatbányászati feladatot lát el, amelynek meghatározott biológiai jelentése van.

Munkámmal sikerült az adott problémára egy lehetséges megoldást megalkotnom, és bízom benne, hogy ezen túlmenően olyan ismereteket, tapasztalatokat felhalmoznom, amelyek jelenleg közvetlenül nem hozzáférhetőek, és ezek a jövőben segítséget nyújtanak az olvasónak bizonyos felmerülő kérdések megválaszolásában, illetve problémák megoldásában.

10. Irodalomjegyzék

- [1] Watson, J. D., Crick, F. H. C.: A Structure for Deoxyribose Nucleic Acid. *Nature*, Vol. 171., 1953, 737-738. old.
- [2] Kovács, J.: Összehasonlító anatómiai előadások I. Sejtten. *Eötvös Kiadó*, 2004
- [3] <http://www.ttk.pte.hu/biologia/genetika/atg/chap11/ch11a.htm>, 2008-01-13.
- [4] Csaba, Gy.: A sejt szerkezete. *Semmelweis Kiadó*, 2003
- [5] <http://www.ttk.pte.hu/biologia/genetika/atg/chap12/ch12a.htm>, 2008-01-13.
- [6] Brookes, A. J.: The essence of SNPs. *Gene*, Vol. 234., 1999, 177-186. old.
- [7] http://www.ornl.gov/sci/techresources/Human_Genome/medicine/assist.shtml, 2008-01-13.
- [8] <http://www.ncbi.nlm.nih.gov/About/primer/bioinformatics.html>, 2008-01-13.
- [9] Pongor, S.: A molekuláris biológia ismeretábrázolási problémái. *Magyar Tudomány*, 2005/4, 418-425. ol.
- [10] Mérő, L.: Új észjárások. A racionális gondolkodás ereje és korlátai. *Tericum Kiadó*, 2001
- [11] Erdős, P., Rényi, A.: On Random Graphs. *Publicationes Mathematicae Debrecen*, Vol. 6., 1959, 290-297. old.
- [12] Barabási, A. L.: Behálózva. A hálózatok új tudománya. *Magyar Könyvklub Rt.*, 2003
- [13] Patthy, L.: A genomkorszak bioinformatikája. *Magyar Tudomány*, 2002/5, 575-581. old.
- [14] <http://www.sanger.ac.uk/HGP/overview.shtml>, 2008-01-13.
- [15] <http://www.hgsc.bcm.tmc.edu/projects/mouse/>, 2008-01-13.
- [16] Gibas, C., Jambeck, P.: Developing Bioinformatics Computer Skills. *O'Reilly & Associates*, 2001
- [17] <http://www.enzim.hu/~szia/bioinformatika/ea03/ea03.htm>, 2008-01-13.
- [18] Needleman, S., Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, Vol. 48., 1970, 443-453. old.

- [19] Smith, T.F., Waterman M.S.: Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, Vol. 147., 1981, 195-197. old.
- [20] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. *Journal of Molecular Biology*, Vol. 215., 1990, 403-410.old.
- [21] Bedel, J., Korf, I., Yandell, M.: BLAST. *O'Reilly & Associates*, 2003
- [22] http://www.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html, 2008-01-13.
- [23] <http://www.ncbi.nlm.nih.gov/BLAST/developer.shtml>, 2008-01-13.
- [24] <http://www.biomart.org/>, 2008-01-13.
- [25] http://www.workflowpatterns.com/patterns/control/advanced_branching/wcp7.php, 2008-01-13.
- [26] <http://taverna.sourceforge.net/>, 2008-01-13.
- [27] <http://kepler-project.org/>, 2008-01-13.
- [28] <http://www.php.net/>, 2008-01-13.
- [29] <http://httpd.apache.org/>, 2008-01-13.
- [30] <http://www.mysql.com/>, 2008-01-13.
- [31] <http://www.mozilla.org/>, 2008-01-13.
- [32] <http://www.bioinformatics.ubc.ca/pegasys/>, 2008-01-17.
- [33] <http://www.trianacode.org/>, 2008-01-07.
- [34] <http://framework.zend.com/manual/en/coding-standard.coding-style.html#coding-standards.inline-documentation>, 2008-01-17.

11. Ábrajegyzék

1. ábra. A sejtmag a sejtben	4
2. ábra. A DNS kettős spirál alakja és a bázispárok elhelyezkedése	5
3. ábra. A gén a DNS egy szakasza, amelyet intron és exon részek alkotnak	6
4. ábra. Számítógép által rajzolt molekula térszerkezet.....	10
5. ábra. Microarray chip.....	10
6. ábra. A Nature 2001. február 15-én megjelent, 409. száma	11
7. ábra. Egyszerű manuális összrendezés.	13
8. ábra. Pontábrázolás.	14
9. ábra. Globális és lokális illesztés.	14
10. ábra. Az aminosavak csoportosítása kémiai tulajdonságaik alapján.	15
11. ábra. A BLOSUM62 mátrix.....	16
12. ábra. A pontszám változása a szekvencia kiterjesztés során és a vágási pont meghatározása.....	19
13. ábra. Konzisztens és inkonzisztens találatok szemléltetése a keresési térben	19
14. ábra. Martview felület.....	22
15. ábra. A Taverna munkafolyamat tervezői felülete.....	26
16. ábra. Eredmények megjelenítése a Taverna végrehajtási nézetében	27
17. ábra. Kliens-szerver architektúra külső Internetes adatbázisok elérésével.....	29
18. ábra. Adatbázis absztrakciós réteg.....	30
19. ábra. A program adattáblái és a közöttük lévő relációk.....	41
21. ábra. A megvalósított rendszer grafikus keresőfelülete.....	42
20. ábra. A felhasználói felület egyes régiói.....	42
22. ábra. A találatok grafikus értelmezése.....	43
23. ábra. A bemutatott példa munkafolyamata	45
24. ábra. A munkafolyamat találatainak megjelenítése	48
25. ábra. A keresőszavakra adott találatok száma az Entrez nukleotid adatbázisban.....	51

12. Abstract

This thesis is related to bioinformatics which is a rarely taught and researched field of interest in Hungary. A short summary of biological definitions can be found in the introduction and besides the necessary basics for further discussion of the problem can give an insight for those who are interested in this topic. Introducing the current specialty I tried to formulate the justification of bioinformatics in the field of sciences and also find cross-disciplines compared to a different kind of approach, which is more acceptable for me. The next section of the paper explains the essential algorithms used by the services, which are applied in this software's workflow. Particularly, giving the outline of the descriptions according to the sequence's alignments of algorithms in the level which has to be mentioned if one wants to understand the problems I am dealing with. The occupied web services are used in most of these algorithms to search in large genomic databases.

In addition, I showed and analyzed in the perspective of usage the necessary tools in solving the problem set by the thesis' aim. These tools are descriptions of web services and online available bioinformatics software, which has applied programming interface. Unfortunately, sources, notes and disciplines about these search oriented web services are not available in Hungarian and this is why I found it important to create a basic level review and a description. The presented methods are built on the entry level exercises but they are trying to achieve a higher level by using more advanced programming and this way creating a workflow system and solving more complex biological workflows.

Under the aegis of engineering, during the work, the planning phase is preceded by giving a planning space for the problem solving. In this section, I showed two different possible solutions capable of doing the set workflow. With a reasonable decision I chose the way for the solution. The realized software is written in PHP containing various components. For example these components are capable of handling MySQL database, by having a database abstraction layer. Additionally making interactions with many online biological databases such as NCBI Entrez or using services like BLAST and BioMart. To clarify the boundaries of the components, all the elements in the software are planned with object oriented paradigm. During the design of the data structure I tended to take advantage from all the capabilities of the relational databases. This way, I created a proper, easily reviewable and improvable structure for storing all the biological and other pieces of information.

The realized software meets the requirements of the original concept by doing data mining in various online databases where this workflow has a certain biological meaning.

I hope that all my work besides creating a possible solution for the given problem, contains such knowledge and experience that are currently not available. Later on, these might give help to the readers by answering some questions and finding solutions connected to this topic.