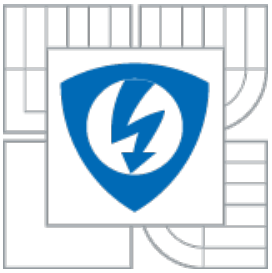
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF BIOMEDICAL ENGINEERING

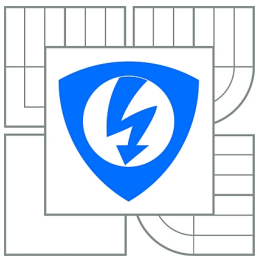
METODY VÍCENÁSOBNÉHO ZAROVNÁVÁNÍ NUKLEOTIDOVÝCH SEKVENCÍ

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE
AUTHOR
Bc. ONDŘEJ TRNĚNÝ

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Denisa Maděránková

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav biomedicínského inženýrství

Diplomová práce

magisterský navazující studijní obor
Biomedicínské inženýrství a bioinformatika

Student:	Bc. Ondřej Trněný	***ID:***	115120
Ročník:	2	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Metody vícenásobného zarovnávání nukleotidových sekvencí

POKYNY PRO VYPRACOVÁNÍ:

1) Proveďte literární rešerši metod pro vícenásobné zarovnávání nukleotidových sekvencí. V rešerši se především zaměřte na moderní přístupy využívající fylogenetiku sekvencí. 2) Na vhodně zvoleném souboru nukleotidových sekvencí vyzkoušejte alespoň tři volně dostupné nástroje pro vícenásobné zarovnávání a výsledky porovnejte. Zhodnoťte vlastnosti, výhody a nevýhody jednotlivých zkoušených nástrojů. 3) Pro vybranou metodu vícenásobného zarovnávání vytvořte pseudokód a podrobný vývojový diagram. 4) V libovolném programovém prostředí implementujte vybranou metodu. 5) Implementovanou metodu vyzkoušejte na zvoleném souboru sekvencí a výsledky porovnejte s výsledky volně dostupných nástrojů.

DOPORUČENÁ LITERATURA:

1] CHAO, Kun-Mao, ZHANG, Louxin. Sequence comparison, theory and methods. London: Springer, 2009, ISBN 978-1-84800-319-4.
[2] SZALKOWSKI, Adam M. Fast and robust multiple sequence alihnment with phylogenyaware gap placement. BMC Bioinformatics, 2012, roč. 13, no. 129.

Termín zadání:	11.2.2013	***Termín odevzdání:***	24.5.2013

Vedoucí práce: Ing. Denisa Maděránková
Konzultanti diplomové práce:

prof. Ing. Ivo Provazník, Ph.D.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Ke správnému pochopení vlastností a účelu biologických sekvencí je nezbytně nutné mít možnost je mezi s sebou porovnávat a ze získaných informací vyvozovat jejich vlastnosti, účel a evoluční historii. K rozšíření již existující báze znalostí mohou velkou mírou přispět metody pro vícenásobné zarovnávaní sekvencí, které umožňují nahlížet na již známá fakta ve zcela novém světle dosud neznámých informací o vztazích mezi často na první pohled nepodobnými sekvencemi. Pro provádění této analýzy proto bylo navrženo několik algoritmů, na jejichž základě následně vznikly programy umožňující komplexní analýzu obsáhlých dat. Jedním z nich je algoritmus progresivního zarovnání, který je v rámci této práce implementován.

Klíčová slova

Matlab, objektové programování, vícenásobné zarovnání, párové zarovnání, ClustalW, ClustalX, T-Coffee, Progresivní zarovnání

Abstract

To be able to understand characteristics and purpose of biological sequences correctly, it is crucial to have a possibility to sort and compare them. Because of this need and to extend existing knowledge pool, numerous methods were proposed. Especially in field of multiple sequences alignment. Methods for multiple sequences alignment may provide various valuable information about sequences which failed to show enough similarity in pairwise alignment. According to this, several algorithms were implemented in various computer applications which provide a way to analyse huge sets of data. One of those, the progressive alignment algorithm, is implemented as a part of this thesis

Key words

Matlab, Object oriented programming, Multiple sequences alignment, Pairwise alignment, ClustalW, ClustalX, T-Coffee, Progressive alignment algorithm

TRNĚNÝ, O. *Metody vícenásobného zarovnávání nukleotidových sekvencí.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 67s. Vedoucí diplomové práce Ing. Denisa Maděránková.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Metody vícenásobného zarovnávání nukleotidových sekvencí jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009Sb.

V Brně dne 24. května 2013 .....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Denise Maděránkové za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne 24. května 2013 .....

podpis autora

Obsah

1 Úvod ... 9

2 Bioinformatické minimum ... 10

2.1 Syntéza proteinů ... 11

2.1.1 Transkripce ... 11

2.1.2 Translace ... 12

2.2 Databáze dat ... 12

2.3 Zarovnávání biologických sekvencí ... 13

2.3.1 Přiřazení ... 13

2.3.2 Globální a lokální zarovnání ... 13

2.4 Vícenásobné zarovnávání sekvencí ... 14

2.5 Role vícenásobného zarovnání ve fylogenetické analýze ... 15

2.6 Skórování vícenásobného zarovnání ... 17

2.7 Skórovací matice ... 17

2.8 Dostupné programy pro vícenásobné zarovnání ... 19

2.8.1 ClustalW ... 20

2.8.2 MATLAB Multiple sequences alignment ... 21

2.8.3 MUSCLE ... 21

2.8.4 T-Coffee ... 22

2.8.5 MSA ... 23

2.8.6 Carrillo-Lipman ... 24

3 Implementace algortimů vícenásobného zarovnání ... 26

3.1 Vícenásobné zarovnání tří sekvencí ... 26

3.1.1 Algoritmus ... 27

3.1.2 Koncepce programu ... 27

3.1.3 Návrh řešení algoritmus vícenásobného zarovnání tří sekvencí ... 28

3.2 Progresivní vícenásobné zarovnání ... 29

3.2.1 Algoritmus pro progresivní zarovnání ... 29

3.3 Koncepce programu ... 30

3.3.1 Pseudokód progresivního vícenásobného zarovnání 31
3.3.2 Třída progresivního vícenásobného zarovnání 35
3.3.3 Návrh řešení – algoritmus progresivního zarovnání 36
3.3.4 Vývojové diagramy a popisy metod progresivního vícenásobného zarovnání 37
3.3.5 Použitý programovací jazyk 43
4 Ověření funkčnosti implementovaných algoritmů a vybraných programů 44
4.1 Sady sekvencí pro testování 44
4.1.1 První sada 44
4.1.2 Druhá sada 45
4.1.3 Třetí sada 45
Tnfsg6 Miscellaneous mouse cDNA sequences 45
Dppa3 Miscellaneous mouse cDNA sequences 45
Gorilla gorilla gorilla whole genome shotgun sequence assembly 45
Homo sapiens CBFB gene 45
Hepatitis B virus ASC1036 nonfunctional core antigen precursor 45
Homo sapiens OPA1 gene 45
4.2 Skórovací parametry algoritmů 46
4.3 Výsledky zarovnání – První sada dat 47
4.3.1 Implementovaný algoritmus pro tři sekvence 47
4.3.2 Implementovaný algoritmus pro progresivní zarovnání 47
4.3.3 Matlab Multialign 48
4.3.4 Clustal W 49
4.3.5 T-Coffee 50
4.4 Výsledky zarovnání – Druhá sada dat 51
4.4.1 Implementovaný algoritmus pro tři sekvence 51
4.4.2 Implementovaný algoritmus pro progresivní zarovnání 51
4.4.3 Matlab Multialign 52
4.4.4 Clustal W 52
4.4.5 T-Coffee 53
4.5 Výsledky zarovnání - Třetí sada dat 54

4.5.1 Implementovaný algoritmus pro progresivní zarovnání 54
4.5.2 Matlab Multialign 55
4.5.3 Clustal W 56
4.5.4 T-Coffee 57
5 Srovnání 60
5.1 Score 60
5.2 Časová náročnost 61
6 Závěr 62

Seznam obrázků

Obrázek 1- Struktura DNA 10
Obrázek 2 - Guanin a cytosin 11
Obrázek 3 - Adenin a thymin 11
Obrázek 4 - Lokální zarovnání sekvencí A a B 13
Obrázek 5 -Globální zarovnání sekvencí A a B 13
Obrázek 6 - Vývojový diagram řídícího programu 28
Obrázek 7 - Distanční matice 29
Obrázek 8 - Guide tree 30
Obrázek 9 - Pseudokód metody multiple_align 31
Obrázek 10 - Pseudokód metody pairwise_align 33
Obrázek 11 - Pseudokód metody create_consensus_seq 34
Obrázek 12 - Struktura objektu pro vícenásobné zarovnání 35
Obrázek 13 - Algoritmus progresivního zarovnání 36
Obrázek 14 - Vývojový diagram metody pairwise 42
Obrázek 15 - Vícenásobné zarovnání, sada dat 1 - implementovaný program 47
Obrázek 16 - Vícenásobné zarovnání, implementovaný program progresivní zarovnání 47
Obrázek 17 - Vícenásobné zarovnání, sada dat 1 - MATLAB multialign 48
Obrázek 18 - Vícenásobné zarovnání, sada dat 1 - ClustalW 49
Obrázek 19 - Vícenásobné zarovnání, sada dat 1 - T-Coffee 50
Obrázek 20 - Vícenásobné zarovnání, sada dat 2 - implementovaný program 51
Obrázek 21 - Vícenásobné zarovnání, sada dat 2 - implementovaný program progresivní zarovnání 51
Obrázek 22 - Vícenásobné zarovnání, sada dat 2 -MATLAB multialign 52
Obrázek 23 - Vícenásobné zarovnání, sada dat 2 -ClustalW 52
Obrázek 24 - Vícenásobné zarovnání, sada dat 2 -T-Coffee 53
Obrázek 25 - Vícenásobné zarovnání, sada dat 3, Algoritmus pro progresivní vícenásobné zarovnání 54
Obrázek 26 - Vícenásobné zarovnání, sada dat 3, Matlab Multialign 55
Obrázek 27 - Vícenásobné zarovnání, sada dat 3, ClustalW 56
Obrázek 28 - Vícenásobné zarovnání, sada dat 3, T-Coffee 59

Seznam tabulek

Tabulka 1 - Přehled první sady sekvencí 44
Tabulka 2 - Přehled druhé sady sekvencí 45
Tabulka 3 - Přehled třetí sady sekvencí 45
Tabulka 4 - Skórovací parametry testovaných algoritmů 46

1 Úvod

V posledních desetiletích dochází k rozmachu poměrně nového vědního oboru známého jako bioinformatika nebo dříve také výpočetní biologie. V současnosti se jedná již o dvě samostatné disciplíny, které se však mohou v některých oblastech do jisté míry překrývat. Za faktem, že bioinformatika nabývá stále více a více na důležitosti, stojí masivní rozšíření snadno dostupných a lehce osvojitelných metod pro získávání informací o biologických sekvencích. Tento vývoj je ještě umocněn masivním snížením nákladů na dostatečně výkonná výpočetní zařízení, která se zároveň stala snadno dostupná. Na jedné straně tu tedy máme dostatek dat čekajících na prozkoumání a klasifikaci a na straně druhé prostředky, které umožňují téměř libovolný druh analýzy v reálném čase. Proto se do popředí zájmu dostávají metody pracující s obsáhlými soubory dat o objemu desítek až stovek sekvencí. Pro zpracování takovýchto souborů se již není možné spokojit s klasickým párovým zarovnáním. Proto se na scéně objevují postupy pro zpracování sad o více než třech sekvencích. Tyto metody jsou souhrnně označovány jako vícenásobné zarovnání, ačkoliv jejich přístup k dané problematice se může významně lišit. Ve všech případech je společným znakemsnaha vyrovnat se s vysokými nároky na výpočetní operace. S každou sekvencí, která je zarovnávána, úměrně vzrůstá potřeba výpočetního výkonu. Nicméně jeden rys mají společný, a to je schopnost odhalit skryté podobnosti v souboru sekvencí, které by při klasickém párovém zarovnání zůstaly bez povšimnutí.

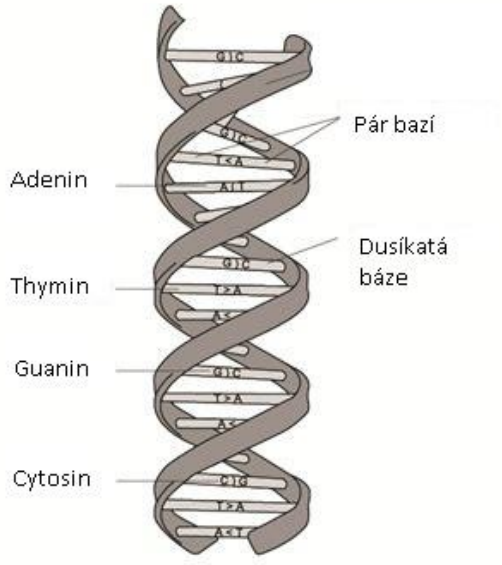
2 Bioinformatické minimum

Deoxyribonukleová kyselina (DNA) představuje základní kámen v procesu přenosu genetické informace v organismech. Účelem DNA je kódování informace pro vývoj buněk, orgánů a celého organismu. DNA byla poprvé pozorována roku 1869 Friedrichem Miescherem. Po jejím objevu následovalo období zkoumání vlastností a složení DNA, které bylo završeno roku 1953 představením její prostorové struktury Jamesem D. Watsonem a Francisem Crickem.

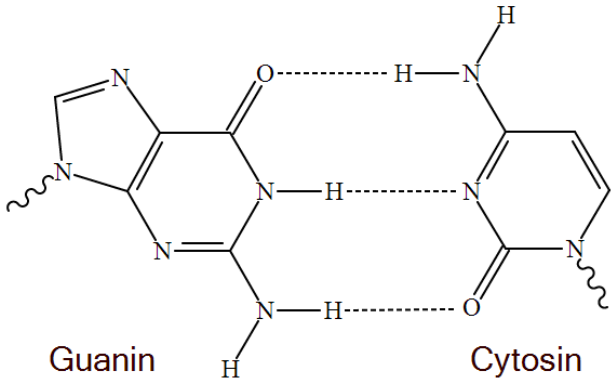
Ačkoliv je DNA společná všem žijícím organismům, nenachází se stále na stejném místě. Obecně však můžeme říci, že v případě eukaryotických organismů je DNA lokalizována v buněčném jádru a v menším množství také v mitochondriích a chloroplastech. Zatímco v případě prokaryotických organismů je volně v cytoplasmě. Uvnitř buněk je organizována v podobě chromozomů. Při buněčném dělení jsou chromozomy v průběhu procesu replikace DNA duplikovány, čímž získá každá buňka svou plnohodnotnou sadu DNA.

Samotná DNA se skládá z řetězce jednotlivých nukleotidů, které jsou složeny z deoxyribózy, fosfátu a dusíkaté báze. Právě dusíkaté báze mají zásadní význam pro informaci, kterou DNA nese. Dusíkaté báze dělíme na purinové (adenin, guanin) a pyrimidinové (thymin, cytosin).

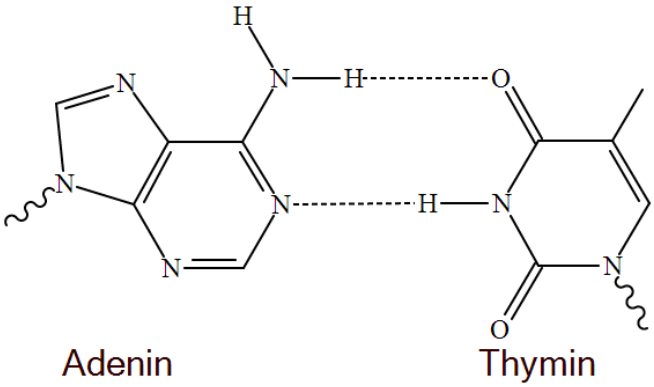
Pro strukturu DNA je zásadní vícevláknová struktura neboli dvoušroubovice. Jedná se o řetězce DNA spojené vodíkovými můstky. Významnou roli v jejím prostorovém uspořádání hrají právě dusíkaté báze, které spojují jednotlivá vlákna podle komplementarity bází. Tedy adenin (A) se páruje s thyminem (T) a guanin (G) s cytosinem (C). Páry jsou spojeny vodíkovými vazbami, jejichž počet je specifický pro každý z nich. Adenin s thyminem jsou spojeny dvěma vodíkovými vazbami a guanin s cytosinem třemi vodíkovými vazbami. [22][23][24][25]



Obrázek 1- Struktura DNA



Obrázek 2 - Guanin a cytosin



Obrázek 3 - Adenin a thymin

2.1 Syntéza proteinů

Účelem DNA je přenos informace pro tvorbu proteinů. I když může DNA řetězec obsahovat velké množství nukleotidů, na samotném procesu tvorby proteinu se podílí pouze zlomkové množství z celkového počtu. Takovéto úseky jsou nazývány exony, zatímco nekódující úseky jsou introny. Proces tvorby proteinů se dělí na dvě stěžejní fáze – transkripci a translaci. [24][25]

2.1.1 Transkripce

První fází překladu genetické informace je takzvaný přepis (transkripce). V tomto procesu za pomoci enzymu známého jako RNA polymeráza dochází k přepisu DNA do podoby RNA. Začátkem je nasednutí RNA polymerázy na začátek genu. Tento úsek genu je nazýván promotor. Následuje iniciace, kdy je dvoušroubovice DNA rozpletena. V dalším kroku je řetězec DNA za pomoci RNA polymerázy komplementárně doplňován nukleotidy, které jsou téměř totožné

s těmi v DNA, ale thymin je nahrazen uracilem. Posledním krokem je terminace, kdy je transkripce ukončena a dochází k uvolnění řetězce RNA.[24][25]

2.1.2 Translace

Translace představuje druhou fázi procesu syntézy proteinů. V této fázi je sestavena primární struktura proteinu podle řetězce RNA získaného v průběhu transkripce. Translace probíhá v ribozomech, kdy se ribozom posouvá po řetězci RNA. V průběhu translace je RNA překládána na aminokyseliny. O jakou aminokyselinu se jedná, rozhoduje kombinace tří po sobě jdoucích nukleotidů, takzvaný kodon/triplet. Translace začíná na úseku RNA uvozeného iniciačním kodonem a končí v místě, kde se nachází stop kodon.[13][24][25]

2.2 Databáze dat

V případě analýzy biologických dat či v našem případě nukleotidových sekvencí, pokud nechceme zkoumat konkrétní sadu sekvencí, je vhodné si experimentální data opatřit prostřednictvím databází. V případě DNA se jedná především o databáze GenBank, EMBL-Bank (European Molecular Biology Laboratory Bank) a DDBJ (DNA Data Bank of Japan). Ve všech třech případech se jedná o moderované databáze, kdy jsou data zveřejňována jen, pokud vyhoví stanoveným požadavkům, jako je zveřejnění v recenzovaném časopise. Z předchozího je zřejmé, že data sekvencí nacházejících se v těchto databázích je možné téměř bez obav použít k experimentům. Výsledky je pak možné kdykoliv reprodukovat, jelikož jsou data veřejně přístupná.[26]

Většina databází se však neomezuje na pouhé skladování dat. Zároveň s obrovským objemem sekvencí je nutné poskytovat i kvalitní a robustní nástroje pro jejich vyhledávání a třídění podle široké palety kritérií. Asi nejznámějším a nejvýznamnějším rozhraním pro vyhledávání je Entrez. Tento systém zastřešuje nejen vyhledávání, ale i portálové aplikace, tudíž umožňuje pohodlné procházení dat prostřednictvím internetu a jejich řazení včetně následné podpory stahování sekvencí v rozšířených bioinformatických formátech jako je FASTA. Entrez je integrální součástí NCBI (National Center for Biotechnology Information). Pod hlavičkou NCBI se pak nachází i GenBank. Kromě vyhledávání většina databází umožňuje provádět přímo online, více či méně komplexní, analýzu vybraných dat. Jako příklad mohou posloužit webové mutace programů ClustalW a ClustalX.[23][26]

2.3 Zarovnávání biologických sekvencí

Párové zarovnání představuje základ pro bioinformatickou analýzu dvou genetických sekvencí. Zarovnání dvou sekvencí je reprezentováno shodou dvou znaků na stejné pozici v obou porovnávaných sekvencích. Sestavené zarovnání potom může poukazovat na příbuznost těchto sekvencí nebo také jejich podobnou funkci.[12][23]

2.3.1 Přiřazení

Nejjednodušším způsobem zarovnání je přiřazení. Porovnávané sekvence jsou přiloženy pod sebe tak, aby byly jejich začátky na stejné pozici. Kvalitu takového postupu potom určuje vypočtené skóre, kdy každému páru přiřadíme předem stanovenou hodnotu pro shodu (match) nebo neshodu (mismatch). Celkové skóre je potom sumou takto získaných hodnot. Zásadní omezení této metody je, že ji nelze použít, když máme nestejně dlouhé sekvence.[12]

2.3.2 Globální a lokální zarovnání

Pro případy, kdy je potřeba zarovnat nestejně dlouhé sekvence, lze použít dvě odlišné metody – globální a lokální zarovnání. Obě metody mají ale své nevýhody. V případě globálního zarovnání se jedná o vkládání mezer (gap) představující deleci, ke které mohlo dojít v průběhu odlišného vývoje porovnávaných sekvencí. A lokální zarovnání se omezuje na přiřazení kratšího úseku, který vykazuje podobnost a tam, kde se sekvence příliš liší, s přiřazením končí. [12]

Mějme tedy pár sekvencí A a B:

A – ACTCAGCTAAACGATCAGACT

B – ATCATACTATCATTT.

```
AGCTAAACGATCA
| | |:|| ||||
ATC-ATACTATCA
```

Obrázek 4 - Lokální zarovnání sekvencí A a B

```
ACTCAGCTAAACGATCAGACT
| |||  |  || |||| : |
A-TCA--T--ACTATCA-TTT
```

Obrázek 5 -Globální zarovnání sekvencí A a B

2.4 Vícenásobné zarovnávání sekvencí

V předchozích odstavcích byla naznačena významnost porovnávání biologických sekvencí mezi sebou. Ovšem ne vždy lze odhalit příbuznost porovnáním pouze dvou sekvencí, resp. ze samotného párového zarovnání nemusí být zřejmá podobnost. Proto bylo navrhnuto množství postupů pro zarovnání více než dvou sekvencí. Tím ovšem proces analýzy nekončí. Je nutné si stanovit kritéria pro zhodnocení kvality výsledného zarovnání, a proto je definováno skóre, které umožňuje rozhodnout o kvalitě provedeného zarovnání. Současné zarovnání několika sekvencí tak představuje velmi významnou součást bioinformatiky a tím s sebou nese i zvýšené nároky na řešení jeho problematiky. Jeho účelem je nalézt biologickou podobnost nebo příbuznost mezi souborem sekvencí. Tento přístup může být použit pro určování konzervativních regionů, genové regulace a evoluční příbuznosti genů nebo proteinů. Škála využití takto získaných informací se neomezuje jen na určování příbuznosti, je možné je využít i pro návrh primerů pro PCR nebo pro stanovení homologních úseků mezi sekvencemi, případně mezi již známými skupinami sekvencí. Rychlost, s jakou se objevují nová sekvenční data vhodná pro analýzu, si vyžaduje stálé vylepšování současných analytických metod, a to nejen v oblasti rychlosti analýzy dat, ale také v přesnosti výsledného zpracování. Vícenásobné zarovnávání staví na základech představených postupy pro zarovnání dvou sekvencí. Zarovnání dvou sekvencí představuje shodu znaků na stejné pozici zarovnávaných sekvencí. Správné zarovnání nukleotidů nebo aminokyselin v sekvencích reprezentuje jejich evoluční příbuznost. Pro určení podobnosti analyzovaných sekvencí jsou tyto přiloženy pod sebe tak, aby obě měly stejně umístěný začátek. Tento postup je nazýván přiřazením. Následně je určena (vypočtena) celková hodnota podobnosti tzv. skóre (score). A to pro každou dvojici vzniklou přiřazením zvlášť. Tyto hodnoty se odvíjejí od předem daných pravidel. Při nejjednodušším stanovení skóre mohou nastat dvě varianty – shoda (match) a neshoda (mismatch).[1][4][8][15]

V případě, že algoritmus využívá přístupu dynamického programování je výsledkem optimální zarovnání. Optimální zarovnání je takové, které má v součtu nejmenší hodnotu vložených mezer a inzercí a delecí. Při zpracovávání nukleotidových sekvencí je vhodné pro přiřazení hodnoty všem párům (přiřazuje kladnou hodnotu) použít některou z dostupným skórovacích matic jako je NUC44 a nepárům (záporná nebo nulová hodnota). Při zarovnání proteinů je problematika náročnější a pro přiřazování hodnot existuje několik typů matic, jakou je například matice PAM (Point accepted mutation). Hodnoty v těchto maticích reflektují frekvence jednotlivých záměn aminokyselin. Často zmiňovaným zástupcem této skupiny matic je PAM 250, která odpovídá 250 mutacím na 100 aminokyselin. Druhou rodinu matic pak zastupují matice typu BLOSUM.[4]

Každý sloupec zarovnání je nazýván zarovnaným párem. Obecně je potřeba, aby sloupec neobsahoval dvě mezery ve sloupci, pro takový případ existuje termín nultý sloupec. Pokud se

rozhodneme dočasně akceptovat existenci nultých sloupců, pak takové zarovnání nazýváme kvazi-zarovnání.[1][3]

Předpokládejme, že máme zadáno S_1, S_2,…, S_m, sekvence, které se skládají z písmen. Vícenásobným zarovnáním těchto sekvencí získáme soubor písmen a mezer o rozměrech m x n, který splňuje podmínku, že žádný sloupec neobsahuje pouze pomlčky. A pokud je odstraníme z řádku i, tak získáme sekvenci S_i o rozměrech $1 \leq i \leq m$. Pro každý pár sekvencí, například S_i a S_j řádky i a j skládají z párového kvazi-zarovnání. Odstraněním všech nultých sloupců získáme párové zarovnání těchto sekvencí. Jedním z často používaných přístupů pro vícenásobné zarovnání sekvencí je přístup založený na algoritmu Needleman-Wunsch. Ačkoliv byl původně určen čistě pro zarovnávání dvou sekvencí, je možné jej s několika změnami aplikovat na libovolné množství sekvencí, což vyústí v matici o N-rozměrech. Bohužel tento jednoduchý způsob s sebou nese zvýšené požadavky na výpočetní čas, které jsou přímo úměrné počtu a délce sekvencí, a tím i celkovému počtu prováděných operací. Vzhledem k náročnosti simultánních výpočtů je tento přístup pro současné zarovnání všech sekvencí používán jen zřídka. Místo toho je možné vypočítat pro všechny zarovnávané sekvence párové zarovnání a až na základě těchto mezi výpočtů sestavit výsledné vícenásobné zarovnání.[1]

2.5 Role vícenásobného zarovnání ve fylogenetické analýze

Vícenásobné zarovnání nemusí nutně představovat konečný výsledek v analýze sekvencí. Naopak jeho výstupy mohou sloužit jako podklady pro další zkoumání zpracovávaného vzorku dat. Jednou z takových možností je jeho využití ve fylogenetické analýze. Jeden ze stěžejních problémů fylogenetiky představuje otázka, jaký je správný přístup pro analýzu dat. Proto je velmi žádoucí mít k dispozici co nejvíce informací o porovnávaných sekvencích. Veškeré metody pro analýzu vztahů mezi daty jsou od základu založeny na zarovnání. Postupem času bylo zjištěno, že výsledek fylogenetické analýzy je více závislý na metodě zvolené k zarovnání zkoumaných sekvencí, než na konkrétním postupu fylogenetické rekonstrukce. Při celkovém pohledu na problematiku toto tvrzení dává smysl, jelikož data, která se analyzují, nejsou pouhé shluky znaků bez větších souvislostí, ale jsou výstupem zvolené metody zarovnání, která tak může významně ovlivnit celkový výsledek.[2][3]

Prvním krokem ve fylogenetické analýze je stanovení homologních částí analyzovaných dat. A právě v tomto okamžiku je vhodné použít některou z metod pro zarovnání sekvencí nukleotidů nebo aminokyselin. Správně zvolené zarovnání umožní identifikovat potenciální homologní úseky zkoumaných sekvencí a tím přispět k potencionálně správnému výstupu například ve formě fylogenetického stromu. Tento první krok je obvykle realizován výpočtem párového zarovnání. Následně je možné tyto domnělé homologní sekvence podrobit některé metodě ze škály fylogenetických analýz. V rámci úspornosti (parsimony) leží zodpovědnost za rozhodnutí,

zda se jedná o správně určené homologní části na kladistické podobnosti znaků z analyzovaných sekvencí. Shodnost prvků nám říká, že ze všech možných způsobů, jakými můžeme sekvence interpretovat, je správný ten, který obsahuje největší podíl shodných znaků. Stupeň shodnosti znaků v souboru dat je potom dán jeho fylogenetickou topologií. Je tedy správné předpokládat, že nejúspornější kladogram sestavený z předchozího zarovnání by měl vycházet z kritérií použitých pro vytvoření tohoto zarovnání. Pokud bychom toto nerespektovali, výsledek by byl založen na odlišných předpokladech než soubor dat, který k němu vedl. Z předchozího dále vyplývá, že nejlepší zarovnání sekvencí je to, které vede k nejúspornějšímu kladogramu. [2][3]

Základním kamenem zarovnání, jak už bylo v předchozím zmíněno, jsou mezery (gaps), které reprezentují evoluční události, kterými mohou být inzerce nebo delece. Vkládání mezer se nelze vyhnout, jestliže soubor dat, která jsou určena k analýze, obsahuje sekvence o různých délkách. Tento postup je nutný pro určení variací v sekvencích. Postup pro přiřazení mezery při zarovnání by měl být zohledněn i v následné fylogenetické analýze založené na tomto zarovnání. Proto by měl být znak pro mezeru považován za pátý znak v souboru znaků (v případě nukleotidových sekvencí). V případě, že při zarovnávání byla mezeře přiřazena nějaká hodnota, tak by tato hodnota měla být brána v potaz i při následné konstrukci kladogramu vycházející z tohoto zarovnání. Je tedy vhodné zmínit, že přiřazení vhodné hodnoty penalizace podstatně ovlivňuje samotné zarovnání a výslednou fylogenetickou analýzu. Počáteční parametry (match, mismatch, gap) je nutné stanovit před samotnou analýzou a tyto důsledně dodržovat vzhledem k tomu, že fylogenetické ohodnocení je také závislé na těchto hodnotách. Tento poznatek vychází z množství různých kombinací analýz, které měly za úkol zjistit, jak velký vliv má volba hodnot parametrů na výsledek (Wheeler, 1995). [2][3]

U téměř každé metody pro porovnání sekvencí je iniciálním krokem párové zarovnání. Většina postupů pro zarovnání sekvencí se snaží aplikovat kritérium (měřítko) podobnosti. Jak již bylo zmíněno výše, existují dvě základní metody pro stanovení podobnosti, a to lokální a globální zarovnání. Velmi dobře se uplatňují například při prohledávání databází (BLAST), ale zde se pracuje s předpokladem, že vyhledáváme sekvence s určitým stupněm podobnosti, které mohou, ale nemusí být homologní. Ve fylogenetické analýze se předpokládá, že zkoumané sekvence vykazují jistou podobnost, resp. jsou příbuzné. Aby tomuto předpokladu bylo učiněno zadost, je vhodné mít co nejvíce informací o zkoumaných sekvencích již před samotnou analýzou, pro tento úkol je globální zarovnání volbou číslo jedna. Vycházíme z toho, že globální zarovnání provádí porovnání po celé délce sekvencí, tedy každý prvek sekvence A porovnává s každým prvkem sekvence B. K dosažení maximální podobnosti mezi sekvencemi je nutné minimalizovat jejich vzájemnou vzdálenost. Vzdálenost znamená počet všech změn, jako jsou substituce, inzerce a delece, které jsou nezbytné pro přiřazení sekvence A sekvenci B. [2][3]

2.6 Skórování vícenásobného zarovnání

Pro jakékoliv dvě sekvence existuje množství způsobů, jak je zarovnat. Je tedy potřeba stanovit si podmínky, podle kterých určíme, které zarovnání je lepší než ty zbývající. Hodnotu, která nám představuje kvalitu zarovnání, označujeme jako score. Obecně můžeme říci, že čím vyšší score je, tím lepší je naše zarovnání. Ale je nutné mít na mysli, že pouze score nemůže popisovat kvalitu zarovnání, je nutné jej doplnit o další ukazatele.

Jako první krok vypočteme score $\sigma(x, y)$ pro každý zarovnaný pár sekvencí $\binom{x}{y}$. V případě, že x nebo y je mezera, potom $\sigma(x, y) = -\beta$. Score σ závisí pouze na obsahu aktuálního zarovnaného páru, nikoliv jejich pozici v sekvenci. Nicméně mohou nastat situace, kdy je vhodné použít score, které zohledňuje pozici páru v sekvencích.

Důležitou součástí skórování párového zarovnání je penalizace za otevření mezery, značená jako α, která je přiřazena každé mezeře v zarovnání – mezera je definována jako místo v řádku zarovnání, značené zvoleným symbolem, které je ukončeno jiným znakem nebo koncem sekvence. Vzhledem ke konstantní penalizaci za otevření mezery se preferuje jedna mezera o konečné délce místo několika osamocených mezer. Tomuto postupu se dává přednost z toho důvodu, že mezera může představovat evoluční událost, ke které došlo v průběhu vývoje sekvence. Score zarovnání je tedy sumou hodnot σ minus α krát počet mezer.

Výběr vhodného skórování, resp. parametrů σ a α je důležitý pro správnost zarovnání. V ideálním případě je zarovnání sestaveno takovým způsobem, že regiony na sekvencích, které nemají žádnou (nebo jen malou) funkci a tedy jejich vývoj probíhá volně, nejsou zarovnány. Naopak úseky sekvencí, které jsou důležité, si zachovají dostatečné odlišení, aby byly při zarovnání zohledněny. Zvolený postup pro skórování tedy ovlivňuje, které úseky jsou označeny jako nezarovnávané a jaké vztahy budou přiřknuty zarovnaným úsekům. Zvolení parametrů score závisí na několika faktorech, mezi které řadíme například evoluční příbuznost druhů, jejichž sekvence porovnáváme.[1][22]

2.7 Skórovací matice

Aminokyselinové substituční matice představují v bioinformatice evoluční modely, které mají za úkol popsat mutace aminokyselin. V praxi to znamená, že každá substituční matice obsahuje hodnocení pro mutaci jedné aminokyseliny na druhou, které je relativní ke stavu, kdy mutace neproběhne. Mezi ty nejznámější patří matice PAM a BLOSUM, které vznikly na základě podkladů získaných z vícenásobných zarovnání. Přičemž se vychází ze statistických metod použitých pro stanovení frekvence všech myslitelných mutací a všech skutečně proběhnuvších mutací.

Prvním zástupcem substitučních matic byla právě PAM, která se objevila v sedmdesátých letech minulého století. Základní matice v řadě PAM1 symbolizuje jednu mutaci na sto aminokyselinových zbytků.[23]

2.8 Dostupné programy pro vícenásobné zarovnání

Ačkoliv existuje množství algoritmů a programů pro sestavení vícenásobného zarovnání, všechny využívají pouze omezeného počtu přístupů, jak zarovnání sestavit. Nicméně dva znaky by měly všechny metody sdílet. Jedná se o snahu o co největší biologickou přesnost a co největší úsporu prostředků, které jsou k dosažení prvního potřebné. Právě potřeba výpočetních prostředků je přímo úměrná nárůstu objemu dat v dostupných databázích. A i když jsou k dispozici stále výkonnější počítače, tak objem dat, který dokáží zpracovat, není neomezený, resp. dat je vždy více, než je možné v rozumném čase zpracovat. Sestrojení biologicky přesného zarovnání tak představuje svého druhu určitou výzvu, při které čas od času selže každá metoda. Proto se setkáváme s rozličnými přístupy k zarovnávání biologických sekvencí. Jak již bylo řečeno, jedním z nejpoužívanějších postupů je heuristický, který využívá párového zarovnání. Jeho nespornou předností je jednoduchá implementace, ale náročnost na výpočet je přímo úměrná počtu prvků v zarovnávaných sekvencích.

Druhým hlavním proudem jsou stochastické a pravděpodobnostní metody. Jejich zástupcem je například využití Skrytých Markovových modelů. Přístup s využitím Skrytých Markovových modelů nahlíží na problematiku vícenásobného zarovnání jako na pravděpodobnostní modely, které všem kombinacím shody, neshody a mezery přiřadí hodnotu pravděpodobnosti. Následně na základě těchto hodnot algoritmus vybere nejpravděpodobnější vícenásobné zarovnání. Jasnou výhodou Markovových modelů je rychlost zpracování porovnávaných sekvencí, ale nevýhodou, kterou s sebou přináší stochastický přístup, je nemožnost dosažení stejného výsledku při opakování. Na druhou stranu představují stochastické metody nový přístup k problematice zarovnání a dá se očekávat, že do budoucna bude jejich důležitost narůstat. Každý přístup k vícenásobnému zarovnání má svá pro a proti, nicméně analýza sekvencí není omezena jednostranným přístupem a existují případy, kdy je použití některého z programů výhodnější, než použití ostatních.[5][22]

2.8.1 ClustalW

Softwarový balík Clustal představuje jeden z nejstarších (1980) a zřejmě i nejpoužívanějších nástrojů pro analýzu biologických sekvencí. Poprvé se objevil v podobě programu napsaného v jazyku Fortran na platformě operačního systému MS-DOS. Myšlenkou, která stála za vznikem tohoto programu, bylo vytvoření multiplatformního nástroje, který bude poskytovat přesné zarovnání v akceptovatelném čase. Toho se podařilo dosáhnout především díky spolupráci biologů a informatiků. [7][8]

Clustal s označením W je vývojovou větví z rodiny programů, která zahrnuje skórovací systém zohledňující pozici prvků v zarovnání a také váhy (odtud označení „W“ z anglického weights) reprezentující skupiny sekvencí. V druhé polovině let 90-tých byly Clustal W a Clustal X(verze s grafickým uživatelským prostředím) nejrozšířenější nástroje pro vícenásobné zarovnání. Jejich předností byla schopnost zarovnávat středně velké soubory dat v přijatelném čase a zároveň byl výstup dostatečně kvalitní, aby nebylo potřeba jej ručně upravovat. V průběhu let se však objevily programy, které poskytovaly přesnější výsledky a byly schopny se lépe vypořádat s velkými soubory dat. Nicméně těžiště použití programů Clustal se začalo postupně přesouvat na internet, kde je možné je použít pro online porovnání dat.[5][8][9]

Významným skokem ve vývoji byla implementace grafického rozhraní, které umožňuje jednoduše procházet zarovnané sekvence a následně provádět změny parametrů a analýzu přímo v jednom okně. Pro lepší vizuální kontrolu jsou sloupce, které si zachovaly pozici, barevně zvýrazněny stejně jako potenciálně špatně zarovnané úseky.[10]

Základní vzorec programu Clustal pro zarovnání sekvencí spočívá v provedení tří nezbytných kroků. Prvním je párové zarovnání všech sekvencí, na jehož základě je vypočtena distanční matice, která určuje rozdílnost pro každý pár sekvencí. Tato fáze je následovaná sestavením do vůdčího stromu (guide tree) z distanční matice, postup pro výpočet se opírá o metodu Neighbor-Joining (NJ). Sekvence jsou následně progresivně zarovnány podle jejich pozice v guide tree, resp. v závislosti na jeho větvích. V případě sestavení guide tree došlo ke změně metody používané pro jeho tvorbu, Neighbor-Joining(byla používána zhruba deset let) byl v nejnovějších verzích nahrazen metodou UPGMA (Unweighted Pair Group Method with Arithmetic Mean). K tomuto kroku vedla především rychlost, se kterou je UPGMA schopná guide tree sestavit. Oba algoritmy, jak UPGMA, tak NJ, mají náročnost na výpočet $O(N^2)$, jasná převaha UPGMA se projeví až při vyšší hodnotě N, řekněme 10000 sekvencí. S tímto souborem dat se algoritmus UPGMA vypořádá zhruba za minutu, zatímco NJ by obdobný výpočet provedl za dobu přesahující hodinu. Jednou z možností, jak snížit časovou náročnost zarovnání, je použití paralelních výpočtů, které využívají pro zarovnání více procesorů. [5] [7] [8] [9]

Vstupní sekvence, které mají být zarovnány, by měly být v podobě některého z všeobecně používaných formátů pro biologická data (FASTA, GenBank, EMBL/Swiss-Prot). Výsledné

zarovnání je potom možné, kromě zobrazení v okně, uložit do souboru v různých formátech (Clustal, PHYLIP, GDE, NEXUS).[7]

2.8.2 MATLAB Multiple sequences alignment

Jedná se o integrovanou funkci vývojového prostředí MATLAB, konkrétně bioinformatics toolbox. Tato utilita implementuje algoritmus pro progresivní vícenásobné zarovnání, které provádí na zadané množině vstupních sekvencí. Prvním krokem vícenásobného zarovnání je výpočet párového zarovnání všech sekvencí. Pro defaultní výpočet skóre je použita Gonnetova skórovací matice, dalšími možnými volbami jsou BLOSUM62 nebo PAM250. Následně je metodou Neighbor-joining vypočten vedoucí (guide) strom, podle kterého jsou odvozeny výsledné variace a evoluční změny v sekvencích.[15]

2.8.3 MUSCLE

MUSCLE je dalším z řady významných programů pro vícenásobné zarovnání sekvencí. Hlavními rysy algoritmu je sestavení guide tree, následované přípravou profilu párového zarovnání. Profil párového zarovnání je nejdříve použit pro progresivní zarovnání a následně pro jeho další úpravy. Prvním krokem k sestavení guide tree je určení vzdáleností mezi jednotlivými páry zarovnávaných sekvencí. MUSCLE implementuje dvě metody pro určování vzdáleností. První je metoda vzdálenosti k-merů, která se používá pro nezarovnané páry a druhou je Kimurova vzdálenost pro zarovnané páry. K-mer je přilehlá podsekvence o délce k. Jiné názvy pro k-mer jsou word nebo k-tice. Metoda k-merů pracuje s předpokladem, že příbuzné sekvence vykazují zvýšený výskyt identických k-merů. Tento postup nevyžaduje zarovnání sekvencí, a proto je možné jej provádět velmi rychle bez velkých výpočetních nároků. V případě, že sekvence jsou párově zarovnané, určíme jejich vzdálenost s Kimurovou korekcí pro vícenásobné substituce nacházející se pohromadě. Konečným krokem této fáze je sestavení guide tree metodou UPGMA z dat distanční matice získané v předchozích krocích.[20][21]

2.8.3.1 Algoritmus

Algoritmus MUSCLE má tři fáze. První je hrubé progresivní zarovnání. Druhá je optimalizované progresivní zarovnání a třetí fází jsou úpravy zarovnání.

První fáze: Na základě guide tree je sestaveno progresivní zarovnání. Pro každý úsek guide tree je sestaven profil na základě vstupní sekvence. Postup po jednotlivých uzlech se provádí v prefixovém pořadí (tedy nejdříve potomek a až pak rodič). Pro každý uzel je následně

vytvořeno párové zarovnání dvou profilů potomků, čímž vznikne nový profil. Tento nový profil je přiřazen zpracovávanému uzlu. Tímto způsobem je sestaveno celé vícenásobné zarovnání.[20]

Druhá fáze: Použití vzdáleností k-merů může vést k jistým nepřesnostem při sestavení guide tree, jak se ostatně u aproximační metody dá předpokládat. Proto je potřeba přehodnotit původní tree za použití Kimurovy vzdálenosti. Nevýhodou tohoto přístupu je nutnost mít sekvence zarovnané, tudíž vzrůstá výpočetní náročnost. Kimurova vzdálenost je vypočítána na podkladech získaných z vícenásobného zarovnání v první fázi. Výsledkem je nová distanční matice. Z ní se opět metodou UPGMA sestaví guide tree. Na základě nového guide tree se provádí progresivní zarovnání. Ale změnou je, že se zarovnávají pouze sekvence, jejichž větve se změnily oproti prvnímu guide tree.[20]

Fáze tři: Úpravy zarovnání začínají od hrany druhého guide tree. Guide tree je pak odstraněním této hrany rozdělen na dva podstromy. Pro část vícenásobného zarovnání, nacházejícího se v obou těchto podstromech, je vypočten profil. Zarovnáním takto vzniklých profilů je sestaveno nové vícenásobné zarovnání. V případě, že toto zarovnání poskytuje lepší skóre, tak je ponecháno. V opačném případě se na něj nebere ohled. Tyto kroky jsou opakovány, dokud se nedosáhne stanoveného zlepšení skóre, anebo není dosaženo přednastaveného počtu cyklů.[20][21]

2.8.4 T-Coffee

T-Coffee (Tree based consistency objective function for alignment Evaluation) je dalším balíkem programů pro vícenásobné zarovnání sekvencí. Umožňuje zarovnávat sekvence protein, DNA a RNA. Kromě samotného zarovnávání je T-Coffee schopen spojovat výstupy z jiných programů, jako je Clustal, Muscle a další. Postup T-Coffee při zarovnávání sekvencí je do značné míry založen na progresivním algoritmu zarovnání. Avšak algoritmus zarovnání byl upraven, aby se předešlo tzv. greedy problémům, kdy dochází k chybnému označení náhodných úseků sekvence jako podobných. Algoritmus postupuje ve třech krocích. Prvním je předzpracování dat – vytvoření knihovny, druhým je párové zarovnání a konečným krokem je úprava knihovny (souboru dat). Knihovna dat obsahuje globální a lokální zarovnání, které jsou používány ke zjištění vzdáleností mezi pírově zarovnanými sekvencemi. Takto upravená knihovna pak slouží k progresivnímu zarovnání zpracovávaných sekvencí.[5][6]

2.8.4.1 Algoritmus T-Coffee

Prakticky se jedná o cyklus vytvoření vícenásobného zarovnání a jeho následnou optimalizaci. Proces sestavení vícenásobného zarovnání má jako vstup globální a lokální zarovnání a výstupem jsou vícenásobná zarovnání, která se dále optimalizují. Optimalizační postup

představuje vytvoření vícenásobných zarovnání, která nejvíce odpovídají knihovně vstupních dat. Tento krok je realizován za použití progresivní metody. Modelovým příkladem může být zarovnání n sekvencí. Pro každý pár je vypočteno globální párové zarovnání programem Clustal W, tento krok je pak proveden i programem Lalign(implementace programu Sim v balíku FASTA), čímž získáme soubor lokálních párových zarovnání. Takto je vytvořena základní knihovna zarovnání, která obsahuje informace o všech párových zarovnání (jejich počet je $n(n-1)/2$). Data uložená v knihovně jsou reprezentována jako skupina spárovaných reziduí. Ovšem ne všechna tato rezidua jsou rovnocená, některá z nich jsou výsledkem zarovnání, která mají vyšší kvalitu a tedy je pravděpodobnější, že jsou správná. Na tuto skutečnost je brán ohled při sestavování výsledného vícenásobného zarovnání, kdy je dána přednost více pravděpodobným reziduím. Stanovení, která rezidua mají přednost, je prováděno podle váhového schématu. [5][6]

2.8.5 MSA

MSA je skupinou nástrojů pro vícenásobné zarovnání biologických sekvencí, dostupná na webu NCBI. Původní verze byla vyvinuta Stephenem Altschulem, Johnem Kececiogluem a Davidem Lipmanem. Účelem tohoto nástroje je vícenásobné zarovnávání sekvencí, které je kardinální pro zisk informací o procesech probíhajících při molekulární evoluci, skládání RNA, genové regulaci nebo vztahu mezi strukturou a funkcí proteinů. Jak již bylo zmíněno v předešlém textu, je většina metod pro vícenásobné zarovnání sekvencí založena na párovém zarovnání jednotlivých sekvencí z datového souboru určeného pro analýzu. Tento přístup, pokud se rozhodneme jej rozšířit na více než dvě sekvence, s sebou nese nevyhnutelné problémy, jakými jsou váhy pro přiřazování hodnot jednotlivým prvkům nebo například penalizace za vložení mezery.[11][22]

Ovšem největším problémem byla, je a zůstává výpočetní náročnost vícenásobného zarovnání. Při potýkání se s touto překážkou alternativní metody pro výpočet zarovnání používají heuristický přístup, nebo se snaží minimalizovat hodnoty pro přiřazené úsekům sekvencí, které přímo nesouvisí s použitým modelem molekulární evoluce. Takovéto zjednodušení ale nevyhnutelně vede k nemožnosti jasně stanovit kvalitu celkového zarovnání produkovaného těmito metodami.[11][22]

2.8.6 Carrillo-Lipman

Použití dynamického programování pro zarovnání n sekvencí vyžaduje pevně stanovený počet operací pro každou buňku ve vzniklé matici o n-rozměrech. Počet takovýchto buněk je závislý na délce sekvencí. Vzhledem k průměrné délce sekvencí je takovýto postup nevhodný pro více než tři sekvence. Nicméně se našel způsob (MSA jej implementuje), jak výrazně snížit zátěž na výpočet. Algoritmus Carrillo-Lipman tak omezuje množství buněk, které je nutné prozkoumat pro sestavení zarovnání. Tato metoda nahlíží na vícenásobné zarovnání, kdy hledáme zarovnání (cestu) v matici o n-rozměrech složenou z párových zarovnání, jako na cestu promítanou na dvourozměrný graf. Je tedy možné vypočíst horní hranici pro ohodnocení projekce optimálního párového zarovnání na konkrétní pár sekvencí. Tato horní hranice ohodnocení pak omezuje počet bodů, kterými může projekce cesty procházet v příslušném dvourozměrném grafu. A tímto dochází i k omezení bodů, kterými může procházet optimální zarovnání. Každá projekce tak vytváří podmnožinu prvků z původní matice, která obsahuje všechny možné kombinace cest pro optimální zarovnání. Průnik těchto podmnožin stále obsahuje všechny tyto cesty. A právě tento průnik je oblastí, kterou by měl algoritmus prohledávat k nalezení optimálního zarovnání. V reálném použití to znamená takovou redukci pozic, které je potřeba prohledat, že je možné efektivně zarovnat v krátkém čase až šest sekvencí.[5][11][17]

2.8.6.1 Omezení shora

Program MSA umožňuje uživateli stanovit požadované omezení shora pro hodnotu zarovnání každého páru sekvencí. Pro každý pár pak program spočte, které části příslušného grafu cesty mohou být použity pro zarovnání tak, aby nedošlo k překročení definovaného omezení. Tímto způsobem v dalším kroku MSA bere v potaz pouze ty části n-rozměrné matice, na které je možné vztáhnout ty části všech grafů cesty, které vyhověly stanovené podmínce. Výsledkem tohoto postupu jsou zarovnání s minimální hodnotou a zároveň jejich graf cesty v prohledávané oblasti. V praxi jsou zvolená omezení téměř vždy větší, než je nezbytné. V takovém případě MSA na základě vlastní analýzy zvolí vhodná omezení. [11][17]

2.8.6.2 Penalizace mezer

Obecně platí, že pokud chceme nalézt z biologického hlediska přijatelné zarovnání, je potřeba penalizovat vkládání mezer. Pro sumu párů (Sum of Pairs) je tato penalizace rovna součtu všech hodnot všech mezer vložených do daného párového zarovnání. Ale dalším zkoumáním se došlo ke zjištění, že striktní dodržování tohoto přístupu vede ke komplikacím v implementaci. Program MSA používá penalizaci odvozenou od přirozené penalizace. Rozdílem je, že v některých případech se cena za vložení mezery adaptabilně mění.[5][11]

2.8.6.3 Váhy párů

Ohodnocení vícenásobného zarovnání založené na SP (suma párů) v některých případech vykazuje nežádoucí vlastnosti. Může nastat případ, kdy se v souboru zarovnávaných sekvencí objeví několik sekvencí, které jsou si velmi podobné. Pokud by měla všechna párová zarovnání stejnou váhu, došlo by k situaci, že příliš podobná zarovnání by ovlivnila výsledné vícenásobné zarovnání a zbylá párová zarovnání by ustoupila do pozadí. Proto je potřeba zajistit, aby byly identické informace zohledněny při dalším postupu, resp. byly vypořádány tak, aby neovlivňovaly negativně výsledné zarovnání. MSA využívá dvě metody pro přidělování vah párovým zarovnáním. Obě tyto metody závisí na evolučním stromu sestaveném pro zarovnávané sekvence. Evoluční strom je sestaven metodou Neighbor-Joining. Konečné slovo má však uživatel, který rozhoduje, zda zarovnávané páry budou mít stejnou váhu nebo ne.[11]

3 Implementace algoritmů vícenásobného zarovnání

3.1 Vícenásobné zarovnání tří sekvencí

Zarovnání tří sekvencí je založeno na modelu párového zarovnání. Mějme sekvence $A = a_1a_2 \dots a_{n1}$, B $= b_1b_2 \dots b_{n1}$ a $C = c_1c_2 \dots c_{n1}$. A předpokládejme, že score $X(x,y)$ je určeno pro každý zarovnaný pár $\binom{x}{y}$. Score zarovnaného sloupce $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ je definováno jako $\emptyset(x,y,z)$ a je možné jej vypočítat jako sumu $X(x,y)$, $X(x,z)$ a $X(y,z)$.

$S[i,j,k]$ představuje score optimálního zarovnání $a_1a_2 \dots a_{n1}$, $b_1b_2 \dots b_{n1}$ a $c_1c_2 \dots c_{n1}$. Při korektní incializaci $S[i,j,k]$ pro $1 \le i \le n_1$, $1 \le j \le n_2$ a $1 \le k \le n_3$ může být score vypočítáno pro následující případy:

$$S[i,j,k] = max \begin{cases} S[i-1,j,k] + \delta(a_i,-,-) \\ S[i,j-1,k] + \delta(-,b_j,-) \\ S[i,j,k-1] + \delta(-,-,c_k) \\ S[i,j-1,k-1] + \delta(-,b_j,c_k) \\ S[i-1,j,k-1] + \delta(a_i,-,c_k) \\ S[i-1,j-1,k] + \delta(a_i,b_j,-) \\ S[i-1,j-1,k-1] + \delta(a_i,b_j,c_k) \end{cases}$$

(3.1.1)[5]

Hodnota $S[n_1,n_2,n_3]$ je potom score optimálního vícenásobného zarovnání sekvencí A, B a C. Sestavená trojrozměrná matice obsahuje $O(n_1,n_2,n_3)$ záznamů. Každý z těchto záznamů představuje maximální hodnotu ze 7 pozic, které danou pozici obklopují.[1][5]

Součástí této práce je samostatný program pro vícenásobné zarovnání. Program je implementován v prostředí Matlab. Na vstupu této aplikace jsou tři nukleotidové sekvence a skórovací parametry, výstupem je pak vícenásobné zarovnání a celkové score. Program je limitován výkonem počítače, na němž je spouštěn. Při kapacitě operační paměti 8GB a předpokladu využití 64 - bitového operačního systému a runtime prostředí Matlab se hranice délky sekvencí, které je možné zarovnat, se pohybují kolem 800 bp. Tento limit je dán postupem použitého algoritmu pro zarovnání. Tento sestavuje 3D matici, tudíž s délkou sekvencí významně narůstají nároky na dostupné systémové prostředky.[5]

3.1.1 Algoritmus

Použitý algoritmus nejprve aplikuje na zadané sekvence postup pro globální zarovnání. Předpokládejme, že máme vstupní sekvence A, B a C. Každá sekvence je porovnána s každou podle klíče A – B. A – C, B - C. Hodnoty pro shodu, neshodu a mezeru získáme ze zadaných skórovacích parametrů. Tímto postupem získáme hodnoty okrajových buněk 3D matice ohodnocení. Zároveň s tím je nutné ve druhé 3D matici cesty uchovávat informace o tom, z které ze sousedních buněk daná hodnota na aktuální pozici přišla. Kódování pro určení cesty obstarávají tři číslice, kdy 1 představuje posun o -1 v daném směru, 0 pak žádný posun.

Druhým krokem je výpočet hodnot vnitřních částí 3D matice ohodnocení. Zde postupujeme podle vzorce

Při výpočtu je potřeba zohledňovat nutné přednosti pro mapování cesty, které jsou v tomto pořadí: 111, 110, 101, 011, 100, 010, 001.

Třetím krokem je potom samotné zarovnání, které začíná v bodě $S[i,j,k]$ a podle matice cesty postupujeme až do bodu $S[1,1,1]$. Interpretace zarovnání pro aktuální sloupec zarovnání podle indexů cesty je následující:[5]

$$111 \dots použití\ aktuálních\ znaků\ A, B, C$$

$$110 \dots použití\ znaků\ A, B\ a\ mezera\ na\ pozici\ C$$

$$101 \dots použití\ znaků\ A, C\ a\ mezera\ na\ pozici\ B$$

$$011 \dots použití\ znaků\ B, C\ a\ mezera\ na\ pozici\ A$$

$$100 \dots použití\ znaku\ A, a\ mezery\ na\ pozici\ B, C$$

$$010 \dots použití\ znaku\ B, a\ mezery\ na\ pozici\ A, C$$

$$001 \dots použití\ znaku\ C, a\ mezery\ na\ pozici\ A, B$$

(3.1.2)[5]

3.1.2 Koncepce programu

Jak již bylo zmíněno výše, program pro jednoduché zarovnání je realizován v programovém prostředí Matlab. Primární část tvoří hlavní obslužná funkce, která provádí inicializaci proměnných a výpočet hodnot spojených se zarovnáním sekvencí. Druhá funkce, volaná z předchozí, se stará o připravení matic potřebných pro sestavení vícenásobného zarovnání. A třetí funkce, volaná nakonec, provádí zarovnání sekvencí do finální výstupní podoby vícenásobného zarovnání.

3.1.3 Návrh řešení algoritmus vícenásobného zarovnání tří sekvencí

Obrázek 6 - Vývojový diagram řídícího programu

3.2 Progresivní vícenásobné zarovnání

Z předchozího je zřejmé, že postup, který byl použit pro sestavení vícenásobného zarovnání, není možné použít pro více než tři sekvence. Použitý algoritmus by v případě více sekvencí vyžadoval příliš velkou míru abstrakce (získali bychom matici o *n* rozměrech, kde $n > 3$), kterou je obtížné dynamickým programováním obsáhnout. Každá sekvence navíc by neúměrně zvyšovala objem kódu potřebný pro její zpracování. Bylo tedy potřeba použít jiný, vhodnější algoritmus.

Jako vhodný kandidát pro tento úkol se jeví metoda progresivního zarovnání. Tento postup je snadno implementovatelný a v rámci mezí i dostatečně rychlý pro zpracování velkých objemů sekvencí. Progresivní přístup pracuje s jednoduchým předpokladem, kdy je prvně zarovnán nejpodobnější pár sekvencí, a další jsou přidávány, až se dostaneme k nejméně podobnému.

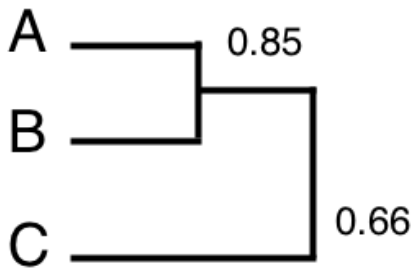
3.2.1 Algoritmus pro progresivní zarovnání

Prvním krokem progresivního zarovnání je sestavení guide tree. Pro sestavení guide tree je potřeba si připravit data. Jako základ nám poslouží distanční matice. Pro určení vzdálenosti mezi sekvencemi je tyto potřeba párově zarovnat a na základě zarovnání stanovit jejich vzdálenost. Vzdálenost mezi párově zarovnanými sekvencemi se určuje jako počet shod děleno délkou zarovnaných sekvencí.[18][19]

A	-		
B	0.85	-	
C	0.51	0.66	-
	A	B	C

Obrázek 7 - Distanční matice

Guide tree je sestaven na základě získané distanční matice, kdy se za nejpodobnější sekvence považují ty, jejichž vzdálenost má nejvyšší hodnotu. Guide tree jako takový nemá žádný fylogenetický význam, tudíž jej není možné použít pro další analýzu nebo hledání evolučních příbuzností.[18][19]



Obrázek 8 - Guide tree

Na základě guide tree je možné začít zarovnávat sekvence. Nejdříve se začne se sekvencemi, které si jsou nejbližší. V našem případě se jedná o sekvence A a B. Jako další v pořadí je sekvence C. Provádíme párové zarovnání, kdy předchozí, již zarovnané, sekvence reprezentuje konsensuální sekvence. Pro výsledné zarovnání je vypočteno celkové score. Jako metoda pro jeho výpočet byla použita Sum of Pairs.[18][19]

3.3 Koncepce programu

Pro aplikaci provádějící vícenásobné zarovnání tří a více sekvencí byla zvolena odlišná strategie, než tomu bylo u programu pro vícenásobné zarovnání tří sekvencí. Mimo použití algoritmu progresivního zarovnání, je hlavním krokem vpřed použití zásad objektového programování. Tento přístup umožňuje pro zarovnávané sekvence vytvořit objekt, který obsahuje všechny atributy a metody potřebné pro provádění akcí nezbytných k vícenásobnému zarovnání. Nespornou výhodou je znuvupoužitelnost jednotlivých metod a jejich pokud možno generické rozhraní, takže je možné je volat v libovolném pořadí, případně upravovat jejich logiku bez ovlivnění funkčnosti celku jako takového.

3.3.1 Pseudokód progresivního vícenásobného zarovnání

Pseudokód představuje formu zápisu postupu algoritmu, která však není vázána na jeden konkrétní programovací jazyk a je zároveň čitelná lidským operátorem. V ukázce pseudokódu jsou uvedeny tři nejdůležitější metody implementovaného programu pro vícenásobné progresivní zarovnání. Pseudokódy všech metod jsou součástí příloh.

3.3.1.1 Pseudokód metody multiple_alignment

```
MULTIPLE_ALIGN (alignment)

 alignment.orig_distance_matrix <- alignment.distance_matrix

for i <- 1 to alignment.seq_number - 1

   alignment <- GET_SEQ_TO_ALIGN(alignment)

   check <- kontrola zda je proměnná alignment.consensus_seq prázdná


   if check je prázdná

      alignment.consensus_seq <- alignment.seq_to_align_01


   alignment.seq01 <- alignment.consensus_seq

   alignment.seq02 <- alignment.seq_to_align_02

   alignment <- INITIALIZE_PAIRWISE(alignment)

   alignment <- PAIRWISE_ALIGNMENT(alignment)


   if i = 1

     alignment.multiple_seq_alignment1 <- alignment.temporary_pair_align1,konec

     alignment.multiple_seq_alignment2 <- alignment.temporary_pair_align2,konec


   else

     alignment <- ADJUST_MULT_ALIGN(alignment)

         alignment.multiple_seq_alignmenti+1 <- alignment.temporary_pair_align2,konec

   alignment = CREATE_CONSENSUS_SEQ(alignment)
```

Obrázek 9 - Pseudokód metody multiple_align

3.3.1.2 Pseudokód metody pairwise_alignment

PAIRWISE_ALIGNMENT(alignment)

n <-délka alignment.seq01 + 1

m <-délka alignment.seq02 + 1

score_matrix <- matice nul o rozměrech m a n

direction_matrix <- matice nul o rozměrech m a n

$direction\_matrix_{1,\ konec}$ <- alignment.insert_symbol

$direction\_matrix_{konec,1}$ <- alignment.delete_symbol

$direction\_matrix_{1,1}$ <- 0

pom <- 0

for i <- 1 to n

    $score\_matrix_{1,i}$ <- pom

    pom <- pom + alignment.gap

pom <- 0

for i <- 1 to m

    $score\_matrix_{i,1}$ <- pom

    pom <- pom + alignment.gap

for i <-2 to m

    if $alignment.seq02_{i-1}$ = A

        sub_seq2 <- 1

    elseif $alignment.seq02_{i-1}$ = C

        sub_seq2 <- 2

    elseif $alignment.seq02_{i-1}$ = G

sub_seq2 <- 3

elseif alignment.seq02$_{i-1}$ = T

sub_seq2 <- 4

for j<-2 to n

if alignment.seq01$_{j-1}$ = A

sub_seq1 <- 1

elseif alignment.seq01$_{j-1}$ = C

sub_seq1 <- 2

elseif alignment.seq01$_{j-1}$ = G

sub_seq1 <- 3

elseif alignment.seq01$_{j-1}$ = T

sub_seq1 <- 4

if score_matrix$_{i-1,j-1}$ + alignment.scoring_matrix$_{sub_seq1,sub_seq2}$ > score_matrix$_{i,j-1}$ + alignment.gap a zároveň score_matrix$_{i-1,j-1}$ + alignment.scoring_matrix$_{sub_seq1,sub_seq2}$ > score_matrixi-1,j + alignment.gap

score_matrix$_{i,j}$ <- score_matrix$_{i-1,j-1}$ + alignment.scoring_matrix$_{sub_seq1,sub_seq2}$

direction_matrix$_{i,j}$ <-alignment.match_symbol

elseif score_matrix$_{i,j-1}$ + alignment.gap > score_matrix$_{i-1,j}$ + alignment.gap

score_matrix$_{i,j}$ <- score_matrix$_{i,j-1}$ + alignment.gap

direction_matrix$_{i,j}$ <-alignment.insert_symbol

else

score_matrix$_{i,j}$ <- score_matrix$_{i-1,j}$ + alignment.gap

direction_matrix$_{i,j}$ <-alignment.delete_symbol

alignment.score_matrix <- score_matrix

alignment.direction_matrix <- direction_matrix

alignment <- PAIRWISE_TRACEBACK(alignment)

Obrázek 10 - Pseudokód metody pairwise_align

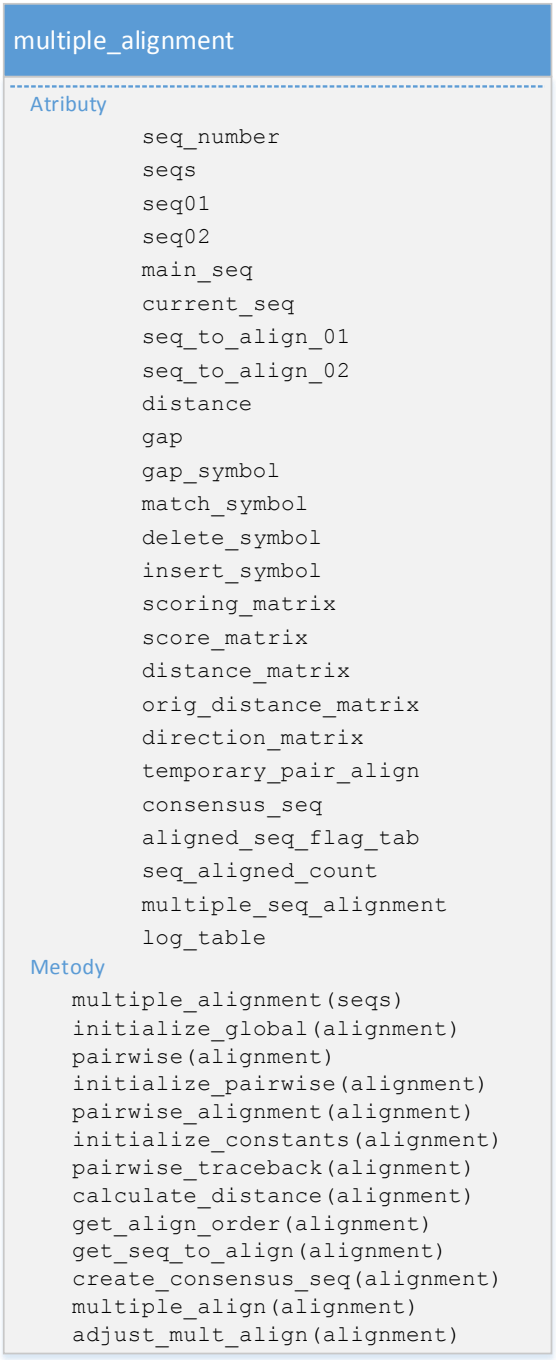
3.3.1.3 Pseudokód metody create_consensus_seq

```
CREATE_CONSENSUS_SEQ(alignment)
for i <- 1 to délka alignment.multiple_seq_alignment1,1

    for j <- 1 to délka alignment.multiple_seq_alignment
        temp_array <- alignment.multiple_seq_alignmentj
        nucleotides <- zřetěz nucleotides a temp_arrayi

    num_a <- suma kolik je v proměnné nucleotides znak A
    num_c <- suma kolik je v proměnné nucleotides znak C
    num_t <- suma kolik je v proměnné nucleotides znak T
    num_g <- suma kolik je v proměnné nucleotides znak G

    value_set <- vytvoř vector z proměnných num_a num_c num_g num_t
    value, position <- najdi hodnotu a pozici maxima vektoru value_set

    switch position
        case 1
            consensus_seq <- zřetěz consensus_seq a A
        case 2
            consensus_seq <- zřetěz consensus_seq a C
        case 3
            consensus_seq <- zřetěz consensus_seq a G
        case 4
            consensus_seq <- zřetěz consensus_seq a T
alignment.consensus_seq <- consensus_seq
```

Obrázek 10 - Pseudokód metody create_consensus_seq

3.3.1.4 Třída progresivního vícenásobného zarovnání

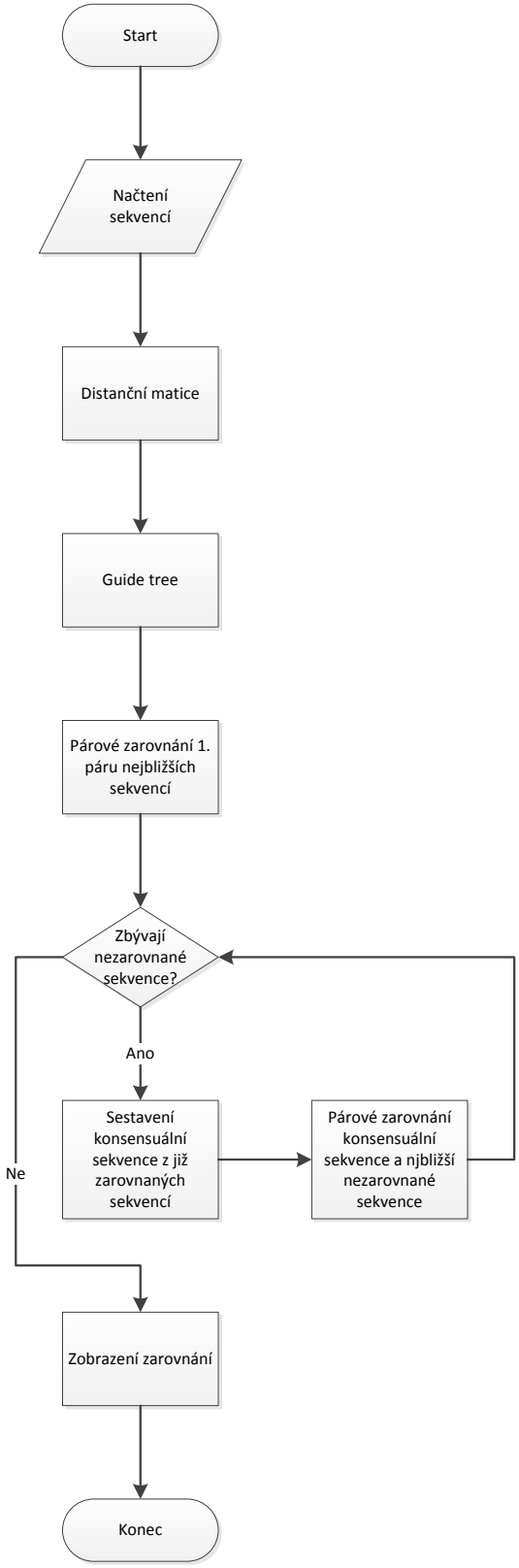
Jádro celého programu představuje definice třídy (multiple_alignment), mající za úkol vytvoření instance objektu, který obsahuje všechno potřebné pro vícenásobné zarovnání nukleotidových sekvencí. Definice třídy na začátku deklaruje atributy dostupné po celou dobu existence objektu. Tyto atributy (aktuálně zarovnávané sekvence, sekvence určené k zarovnání, symboly pro substituce, distanční matice atd.) umožňují uchovávat různé informace, k nimž se přistupuje při volání jednotlivých metod. Díky jejich deklaraci v definici třídy se jedná o globální atributy. Globální atributy na rozdíl od těch lokálních, které se vyskytují v jednotlivých metodách, nejsou vymazány poté, co daná metoda provede svůj zdrojový kód. Počet globálních atributů je vhodné omezit na nejmenší nutné množství, protože jsou udržovány v paměti po celou dobu běhu aplikace. Nicméně pokud chceme s některými daty pracovat i v jiných metodách, tak se jejich použití nevyhneme. Proto je také vhodné, pokud je to možné, v co největší míře používat lokální proměnné, díky kterým můžeme významně snížit náročnost programu na systémové prostředky.



Obrázek 12 - Struktura objektu pro vícenásobné zarovnání

Druhou část definice třídy tvoří deklarace jednotlivých metod. Metody jsou objekty (samostatné úseky kódu), které mají jasně definované vstupy a výstupy. Je možné je libovolně kombinovat, protože jejich interface je uniformní. Každá z nich má na vstupu a výstupu instanci objektu, což jim umožňuje mít okamžitý přístup ke všem atributům a metodám dané instance. Celé zarovnání je realizováno prostřednictvím volání jednotlivých metod, přičemž jsou některé z nich použity opakovaně.

3.3.2 Návrh řešení – algoritmus progresivního zarovnání



Obrázek 13 - Algoritmus progresivního zarovnání

3.3.3 Vývojové diagramy a popisy metod progresivního vícenásobného zarovnání

Následující část obsahuje podrobný popis postupů a operací prováděných metodami implementovaného progresivního zarovnání. Výjimkou je metoda pairwise, která obsahuje i přiložený vývojový diagram pro ilustraci. Podrobné vývojové diagramy všech implementovaných metod se nacházejí v přílohách.

3.3.3.1 Metoda konstruktoru multiple_alignment

První metoda, která se volá v průběhu aplikace vícenásobného progresivního zarovnání, je konstruktorem třídy. Jejím úkolem je vytvořit instanci třídy vícenásobné zarovnání – objekt. V rámci konstruktoru je naplněn globální atribut, který uchovává sekvence určené k zarovnání. A dále jsou volány dvě metody obsluhující inicializaci atributů nezbytných pro správný běh programu. Jedná se o metody initialize_constants a initialize_global. Jak napovídá název první metody, tato provádí přípravu konstant. Tyto atributy jsou neměnné po celou dobu existence objektu. Druhá metoda potom připravuje základní atributy, jako jsou countery a logovací tabulky.

3.3.3.2 Metoda initialize_constants

Metoda initialize_constants obstarává naplnění globálních atributů – konstant. Jedná se o proměnné, které jsou v průběhu celé aplikace neměnné a používané na několika místech, proto je výhodné je deklarovat globálně, aby k nim měly přístup veškeré metody třídy. Jde především o konstanty penalizace mezery, skórovací matice, symbolů pro shodu, deleci, inzerci a mezeru.

3.3.3.3 Metoda initialize_global

Initialize_global je metodou pro vytvoření globálních atributů, které jsou po dobu běhu aplikace hojně používány a je výhodné je v rámci jednorázového úkonu předpřipravit již v rámci volání konstruktoru. Jde především o výpočet počtu sekvencí ke zpracování a logovací tabulku pro určení, které sekvence již byly zpracovány. Tyto globální atributy se mohou v průběhu zpracování měnit.

3.3.3.4 Metoda calculate_distance

Další v řadě metod je metoda pro výpočet podobnosti sekvencí. Na základě právě zarovnaného páru sekvencí je určena délka zarovnání. Následuje předání zarovnaného páru do lokálních proměnných. Tyto lokální proměnné jsou porovnány vůči sobě. Výsledkem je počet shod v zarovnání. Tento počet je podělen celkovou délkou zarovnání, čímž je získána podobnost. Tato hodnota je ve finálním kroku předána do globálního atributu.

3.3.3.5 Metoda get_align_order

Další pomocná metoda, která na základě globálního atributu pro vypočtenou podobnost zarovnaného páru sekvencí aktualizuje distanční matici. Distanční matice je aktualizována na pozici odpovídající sekvencím, které byly předlohou pro párové zarovnání. Tímto způsobem je při další práci s distanční maticí možné dohledat hodnotu podobnosti pro každý pár sekvencí.

3.3.3.6 Metoda mult_align

Pro sestavení vícenásobného progresivního zarovnání byla implementována řídící metoda. Tato se stará o volání konstruktoru multiple_alignment, metody pro určení podobnosti pairwise a metody pro vícenásobné zarovnání multiple_align a jako poslední provádí zobrazení výsledného zarovnání.

3.3.3.7 Metoda sum_of_pairs

Po dokončení vícenásobného zarovnávání sekvencí je potřeba tyto ohodnotit. Jedním z kritérií pro takové hodnocení je právě score. Jeho výpočet probíhá metodou sum of pairs. Kdy je pro každý sloupec výsledného zarovnání vypočtena jeho celková hodnota, a to výpočtem pro všechny páry nacházející se v daném sloupci. Hodnoty jsou vzaty ze skórovací matice a hodnoty pro penalizaci mezery. Cyklus metody prochází všechny sloupce a jeho výsledkem je celkové score, pro výsledné vícenásobné zarovnání.

3.3.3.8 Metoda initialize_pairwise

Před samotným párovým zarovnáním je potřeba připravit globální atributy instance objektu pro další zpracování. Jedná se o inicializaci matice hodnot, kterou je nutné vždy aktualizovat podle délky aktuálně zarovnávaného páru sekvencí. Stejně tak je inicializována matice cesty, která slouží ke zpětnému trasování zarovnání. Následují dva po sobě jdoucí cykly, které plní nultý sloupec a nultý řádek počátečními hodnotami. V případě matice hodnot jsou to hodnoty odvozené od penalizace mezery a u matice cesty se jedná o symboly pro to, odkud hodnota nacházející se na stejné pozici v hodnotové matici přišla.

3.3.3.9 Metoda pairwise_alignment

Metoda pairwise_allignment je integrální částí úseku programu, který má na starosti párové zarovnání. V rámci této metody dochází k porovnávání jednotlivých sekvencí vůči sobě, nukleotid po nukleotidu. Nejprve jsou inicializovány nezbytné lokální proměnné, jako jsou matice hodnot a cesty. Následuje cyklus procházející první sekvencí vždy po jednom znaku. Pro danou pozici se stanoví, jaký nukleotid se na ní nachází. Podle takto určeného nukleotidu je do lokální proměnné přiřazena souřadnice, která odpovídá pozici daného znaku ve skórovací matici. Po tomto úseku vnitřní logiky je prováděn vnořený cyklus, který provede totéž pro druhou

zarovnávanou sekvenci. Pohybujeme se tedy ve skórovací matici po sloupcích. Další zpracování nadřazeného cyklu provádí výběr maximální hodnoty pro danou pozici vzhledem k tomu, zda se jedná o hodnotu zleva (inzerce), shora (delece) anebo diagonální (shoda/neshoda). V případě, že se jedná o shodu, tak vždy upřednostníme právě tuto hodnotu. Výsledek rozhodování je uložen na aktuální pozici v hodnotové matici a symbol značící, jakým způsobem byla hodnota vybrána, je uložen do matice cesty. Následně jsou aktualizovány příslušné atributy objektu z lokálních proměnných pro matice hodnot a cesty. Posledním krokem je volání metody pairwise_traceback, která na základě získaných hodnot provede zpětné trasování průchodu maticí hodnot a dokončí tak párové zarovnání.

3.3.3.10 Metoda pariwise_traceback

Poslední metodou vztahující se přímo k párovému zarovnání je pairwise_traceback. Úlohou této metody je zpětně trasovat průchod maticí hodnot a na tomto základě vytvořit finální párové zarovnání. První část logiky opět inicializuje lokální proměnné, aby nebylo zasahováno do globálních atributů objektu. Je vypočtena délka zarovnávaných sekvencí, na jejich základě se vytvoří array, do něhož bude zapisováno výsledné zarovnání. Délka array je stanovena jako délka té nejdelší ze zarovnávaných sekvencí. Následuje krok inicializace pomocných proměnných, které umožňují stanovit pohyb po matici hodnot a cesty. Další částí je samotné jádro metody, které tvoří while cyklus. V jeho rámci je podmínkou vyhodnocena aktuální hodnota na pozici v matici cesty. Jsou možné tři výsledky. Prvním je, že hodnota na pozici přišla diagonálně. V tomto případě se v následujícím průběhu cyklu posuneme v matici cesty opět diagonálně a do proměnné pro zarovnání zapíšeme nukleotidy z každé sekvence, odpovídající této pozici. Pokud hodnota v matici cesty odpovídá znaku pro inzerci, tak do prvního řádku proměnné zarovnání vložíme znak pro inzerci a do druhého příslušný nukleotid. V dalším průběhu cyklu se posuneme směrem doleva o jednu pozici. Poslední možnost představuje hodnota, která přišla shora, tedy delece. Postup je analogický k inzerci, jen do prvního řádku proměnné zarovnání zapíšeme nukleotid a do druhého symbol mezery. Celý cyklus je prováděn, dokud se nedosáhne levého horního rohu matice cesty, což znamená, že se ocitáme na počátku sekvence a můžeme cyklus ukončit. Takto získané zarovnání zapíšeme do globálního atributu instance objektu.

3.3.3.11 Metoda multiple_align

Samotné vícenásobné zarovnání je realizováno prostřednictvím obslužné metody, která provádí všechny úkony s tím spojené. Využívá znovu použitelnosti objektových metod v podobě procesů párového zarovnání, které je základem zarovnání vícenásobného a zároveň integruje funkcionalitu exkluzivní pouze pro vícenásobné zarovnání. Stejně jako je tomu u předchozích metod jsou nejdříve inicializovány lokální proměnné a zálohována originální distanční matice,

protože v průběhu zpracování bude upravována. V dalším kroku je zavolána metoda, která na základě distanční matice určí sekvence k zarovnání. Dále je zkontrolováno, zda je atribut pro konsenzuální sekvenci prázdný. Pokud je prázdný, je naplněn znaky první sekvence určené k zarovnání. Globální atribut objektu je poté naplněn znaky z proměnné pro konsenzuální sekvenci. Dalším krokem je zavolání metody pro inicializaci párového zarovnání, která je následována metodou pro párové zarovnání. Pokud se jedná o první běh cyklu, tak jsou do globálního atributu pro uchování vícenásobného zarovnání uloženy obě párově zarovnávané sekvence, v ostatních případech jsou již zarovnané sekvence upraveny podle mezer vložených do zarovnané konsenzuální sekvence. Úprava zarovnaných sekvencí se provádí prostřednictvím metody adjust_mult_align. Po tomto kroku je k souboru zarovnaných sekvencí přidána v pořadí druhá sekvence párového zarovnání. Finálním krokem celé logiky a cyklu je volání metody pro vytvoření konsenzuální sekvence, která bude použita v příštím běhu cyklu.

3.3.3.12 Metoda get_seq_to_align

Pro získání sekvencí, které budou zarovnávány je implementována metoda get_seq_to_align. Prvním krokem je naplnění lokální proměnné z globálního atributu pro distanční matici. Zároveň je inicializována proměnná pro průchod while cyklem. Na začátku cyklu je stanovena maximální hodnota v distanční matici. Poté je pro tuto maximální hodnotu zjištěna její pozice v sekvenci. Získané indexy označují, kterým sekvencím tato hodnota náleží. Tímto způsobem jsou získány sekvence s největší podobností. Vzhledem k tomu, že sekvence jsou již zaznamenány, je možné tuto hodnotu nahradit nulou, takže při dalším volání metody nejsou znovu vybrány. V další části je prohledána logovací tabulka, zda už některá ze získaných sekvencí byla někdy předána k vícenásobnému zarovnání. Pokud tomu tak bylo, jsou obě sekvence předány do globálních atributů objektu. Záznam v logovací tabulce je pro příslušnou sekvenci doplněn o příznak zpracování.

3.3.3.13 Metoda create_consensus_seq

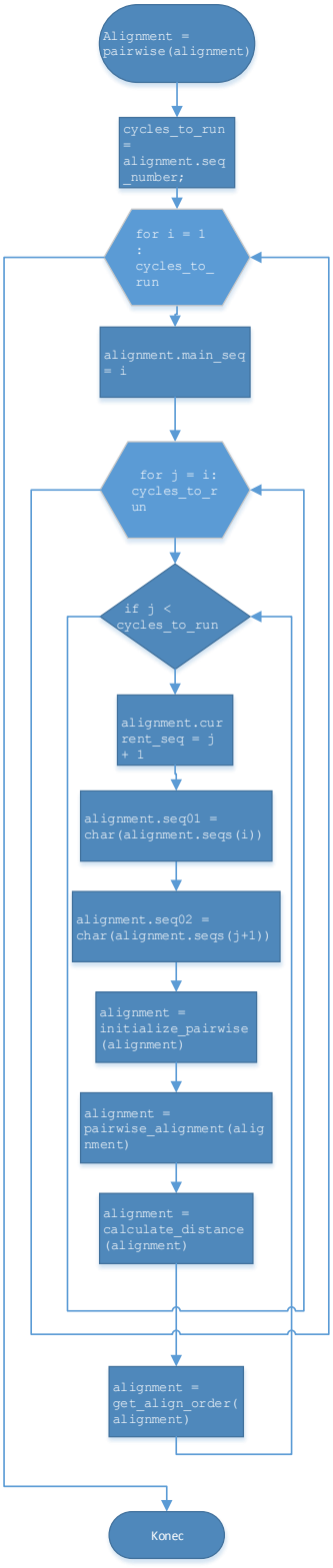
K sestavení konsenzuální sekvence byla implementována metoda create_consensus_seq. Její hlavní částí je cyklus procházející již vytvořené dočasné vícenásobné zarovnání a to právě tolikrát, kolik již bylo zarovnáno sekvencí. Prvním krokem cyklu je inicializace proměnné, do které jsou uloženy všechny nukleotidy, náležící právě zpracovávanému sloupci zarovnání. Tato proměnná je plněna v průběhu vnořeného cyklu. Poté následuje sečtení výskytu jednotlivých nukleotidů ve sloupci. Na základě těchto součtů je vybrán nukleotid, který se vyskytuje nejčastěji (v případě shodného počtu jsou váhy jednotlivých nukleotidů v tomto pořadí: A, C, G, T). Po získání potřebné hodnoty je na základě podmínky switch doplněn nukleotid s nejčastějším výskytem na patřičnou pozici do lokální proměnné, která obsahuje konsenzuální sekvenci. Posledním úkonem metody je aktualizace globálního atributu právě z této lokální proměnné.

3.3.3.14 Metoda adjust_mult_align

Při sestavování vícenásobného zarovnání je potřeba kontrolovat a průběžně upravovat již zarovnané sekvence. Tento postup je nutný vzhledem k tomu, že do konsenzuální sekvence mohou být zanášeny mezery. Tudíž je nutné zpětně ověřit, zda se nezměnila délka zarovnání a v případě potřeby tento problém reflektovat. Metoda, která implementuje tuto logiku, nejdříve přesune zarovnanou konsenzuální sekvenci do lokální proměnné. Následně je určena pozice mezer v konsenzuální sekvenci. V dalším kroku se kontroluje, zda vůbec byly nějaké mezery do konsenzuální sekvence vloženy. Pokud nebyly, tak se ukončí zpracování metody. V opačném případě jsou inicializovány lokální proměnné pro případné rozdělení sekvence na dvě části, aby mezi ně mohla být vložena mezera, reflektující změny, které proběhly v sekvenci konsenzuální. Následuje cyklus procházející již zarovnané sekvence. Každá sekvence určená pro úpravu je načtena do lokální proměnné. Vnořený cyklus pak prochází vektor, ve kterém jsou uloženy pozice vložených mezer. Pro jistotu, že vše proběhne, tak jak má, jsou vyprázdněny proměnné pro rozdělenou sekvenci. Dalším krokem zpracování je podmínka, která musí řešit tři možnosti: mezera se nachází na počátku sekvence, mezera se nachází na konci sekvence a mezera se nachází mezi dvěma úseky sekvence. V prvním případě je celá sekvence posunuta o jeden znak a mezera je vložena na začátek. V případě druhém je sekvence rozdělena a mezera je vložena mezi první úsek a poslední znak sekvence. Ve třetím případě je sekvence rozdělena na dvě části a mezera je vložena mezi ně. Takto upravená sekvence je poté vložena na svou pozici v globálním atributu objektu.

3.3.3.15 Vývojový diagram metody pairwise

Prvním krokem nutným pro vícenásobné progresivní zarovnání sekvencí je párové zarovnání každé sekvence s každou, aby bylo možné vypočíst jejich podobnost. Pro tento úkol byla vytvořena metoda pairwise, která provádí všechny akce spojené se zarovnáním. Po každém párovém zarovnání je vypočtena podobnost mezi sekvencemi a uložena do průběžně vytvářené distanční matice. V praxi to znamená stanovit, kolik běhů cyklu vnitřní logiky metody bude potřeba. Tento údaj je stanoven na základě počtu nukleotidových sekvencí určených k zarovnání. Pročež je aktuální sekvence v průběhu vnořeného cyklu zarovnána se zbývajícími sekvencemi z datového souboru. Na konci každého průchodu vnořeným cyklem je spočtena podobnost zarovnaného páru a aktualizována distanční matice instance objektu o nový údaj.



Obrázek 14 - Vývojový diagram metody pairwise

3.3.4 Použitý programovací jazyk

Jako programovací jazyk pro implementaci algoritmů vícenásobného zarovnání byl zvolen Matlab (Matrix Laboratory). Jedná se o skriptovací jazyk čtvrté generace, jehož předobrazem byl programovací jazyk FORTRAN. Matlab a jeho vývojové prostředí je vhodné pro operace s maticemi a číslicové zpracování, což vzhledem k převážné práci implementovaných algoritmů s maticemi značně usnadňuje implementaci. Dalším faktorem pro zvolení Matlabu je škála jeho rozšíření (toolboxes), která dále zvyšují jeho využitelnost. V rámci této práce je nejvíce ceněným rozšířením Bioinformatický toolbox, který implementuje řadu algoritmů a postupů používaných v bioinformatice. A v neposlední řadě představuje přínos možnost používat zásady objektového programování, které umožňují rychle vytvářet robustní kód. Nedochází tak k situaci klasického procedurálního programování, kdy při větším množství kódu se tento stává rychle nepřehledným a obtížně udržovatelným, nemluvě pak o dalším rozšiřování a úpravách. V rámci objektového přístupu se výsledný program skládá z jednotlivých objektů, které jsou libovolně kombinovatelné díky uniformnímu interface. Není pak problémem přidat další metodu pro zpracování dat, aniž by to negativně ovlivnilo celek jako takový.

Nicméně Matlab neoplývá jen výhodami. Mezi neduhy je možné řadit problematickou práci s přidělováním paměti, kdy po vyčerpání přidělených prostředků RAM dochází k pádu aplikace místo toho, aby se toto řešilo odkládáním dat na disk. A jako nejpalčivější problém je možné vnímat absenci schopného debuggeru, který by umožňoval komfortně ladit program. V praxi se setkáváme s neohrabaným přístupem k aktuálně používaným proměnným, velmi omezenými možnostmi přechodu mezi volanými metodami nebo nemožností skoku k instrukci. Poslední jmenovaný problém, tak uživatele nutí stále znovu procházet debuggovacím režimem od začátku spuštění aplikace místo toho, aby bylo možné jeden krok opakovat vícekrát v rámci jednoho režimu.

4 Ověření funkčnosti implementovaných algoritmů a vybraných programů

4.1 Sady sekvencí pro testování

Pro vícenásobné zarovnání byly vybrány tři sady dat, první dvě po třech sekvencích. Třetí sada pak obsahuje sekvencí sedm a je na ní demonstrován algoritmus pro progresivní vícenásobné zarovnání. První sada obsahuje dvě lidské sekvence a jednu z myši domácí. Druhá sada potom Obsahuje sekvenci lidskou, gorilí a sekvenci makaka. Třetí sada obsahuje tři sekvence myši domácí, dvě sekvence lidské a po jedné sekvenci od gorily a viru žloutenky typu B. Všechny sekvence byly získány z volně dostupné nukleotidové databáze NCBI (http://www.ncbi.nlm.nih.gov/).

4.1.1 První sada

Tabulka 1 - Přehled první sady sekvencí

Název	Délka sekvence[bp]	Kód NCBI
Mus musculus DNA-directed RNA polymerases I, II, and III subunit RPABC4	177	XM_003689219.1
Homo sapiens microRNA 203	110	NR_029620.1
Homo sapiens mitochondrial DNA polymerase gamma (POLG) gene	114	AY377897.1

4.1.2 Druhá sada

Tabulka 2 - Přehled druhé sady sekvencí

Název	Délka sekvence[bp]	Kód NCBI
Macaca mulatta DNA for amylin	111	X59792.1
Human apolipoprotein B mRNA editing enzyme complex-1	111	U72890.1
Gorilla apolipoprotein B gene	159	M59320.1

4.1.3 Třetí sada

Tabulka 3 - Přehled třetí sady sekvencí

Název	Délka sekvence[bp]	Kód NCBI
D15Roc9 Mus musculus NFS/N-sld Genomic DNA	175	BV725426.1
Tnfsg6 Miscellaneous mouse cDNA sequences	178	G73539.1
Dppa3 Miscellaneous mouse cDNA sequences	149	G73534.1
Gorilla gorilla gorilla whole genome shotgun sequence assembly	300	FN579877.1
Homo sapiens CBFB gene	200	AB473218.1
Hepatitis B virus ASC1036 nonfunctional core antigen precursor	150	AF528225.1
Homo sapiens OPA1 gene	180	AF416919.1

4.2 Skórovací parametry algoritmů

Skórovací parametry pro jednotlivé programy odpovídají jejich defaultnímu nastavení

Tabulka 4 - Skórovací parametry testovaných algoritmů

Algoritmus/Program	Match/Skórovací matice	Mismatch	Gap
Implementovaný algoritmus pro zarovnání tří sekvencí	4	-2	-8
Implementovaný algoritmus progresivního zarovnání	jednotková matice		-4
Matlab Multialign	Gonnetova matice		defaultní
Clustal W	IUB		10
T-Coffee	IUB		0

4.3 Výsledky zarovnání – První sada dat

4.3.1 Implementovaný algoritmus pro tři sekvence

Čas zarovnání: 14,7494 s

Score: 320

```
ATGGACGCTCAGAAAGATGTTCAACCACCAAAGCAGCAGCCAATGATATATATTTGTGGAGAGTGTCACACCGAAAA
-----C-C-CAGCGA--CGGGCAGCGGCGGCGGCGGCAG-CA--GC-A-GCA---GCAG-CA--G-CA-GCAG--CA
GTGTT-G----G-G-GAC--TC-GC-GC-GCTG-GG--TCCAGTGGT-T-C--TT--A-ACAGT-T--C----AACA


TGAGATAAAGTCCAGGGATCCCATCAGATGCAGAGAATGTAGATACAGAATAATGTACAAGAAAAGGACTAAAAGATTG
-GC-A-ACAG--CA--G---CC-TCAGCAGCCG-CAA-GT-G---C----T-A--TCCT-----CGG----AG-GGCGG
---G-T----TC--TG--T---A---GC-GC--A--AT-T-G-T---GAA-A-TGTT-TAGGACC--ACTA-GA-CCCG


GTGGTTTTTGATGCTCGATGA
GCAG-CTGCG--GCACAAC--
GCGGGC-GCGGCG-ACAGCGA
```

Obrázek 11 - Vícenásobné zarovnání, sada dat 1 - implementovaný program

4.3.2 Implementovaný algoritmus pro progresivní zarovnání

Čas zarovnání: 0.73524 s

Score: 12

```
-----------GT--GTTGGGGACTCGCG-CGCTGGGT-CCA--GT-GGT-TCTTA---ACAG---TT--C--AACA-GT-
-----C-C-CAGC--GACGGGCAGCGGCGGCGGCGGCA-GCA--GC-AGC-AGC-A---GCAG---CAG-C--AGCA-GC-
ATGGACGCTCAGAAAGATGTTCAACCACCAAAGCAGCAGCCAATGATATATATTTGTGGAGAGTGTCACACCGAAAATGAG

T--CTG-T-AGCG--CA-A-TTGT----GA-AA--T-GTTT-AGGACCACT-A---GACCCGG-CGGGC-GC--GGCGAC-
A-ACAG-C-AGCC-TCA-G-CAGC--C-GC-AA-GT-GCT--AT--CCTCG-----GAG--GG-CGGGCAGC--TGCGGC-
ATAAAGTCCAGGGATCCCATCAGATGCAGAGAATGTAGATACAGAATAATGTACAAGAAAAGGACTAAAAGATTGGTGGTT

----A-GCG--A---
----A--CA--AC--
TTTGATGCTCGATGA
```

Obrázek 12 - Vícenásobné zarovnání, implementovaný program progresivní zarovnání

4.3.3 Matlab Multialign

Čas zarovnání: 0,0719 s

Score:

```
AT--------------GGACGCTCAGAAAGATGTTCAACCACCA-AAGCAGCAGCCAATG-ATATATATTTG-TGGAGAGTGTCACACCG
CCCAGCGACGGGCAGCGGCGGCGGCGGCAGCAGCAGCAGCAGCAGCAGCAGCAGCAACAGCAGCCTCAGCAG---------------CCG
GT---------GTTGGGGACTCGC----------------------GCGCTGGGTCCAGTG-GTTCTTAACAGTTCAACAGTTCTGTAGCG


AAAATGAGATAAAGTCCAGGGATCCCATCAGATGCAGAGAATGTAGATACAGAATAATGTACAAGAAAAGGACTAAAAGATTGGT
CAAGTGCTA-----TCCTCGGA---------------GGGCGGGCAGCTGCGGCACAAC----------------------------
CAATTGTGA-AATGTTTAGGA---CCACTAGACCCGGCGGGCGCGGCGACAGC---------------------------------


GGTTTTTGATGCTCGATGA
--------------------
------------------GA
```

Obrázek 13 - Vícenásobné zarovnání, sada dat 1 - MATLAB multialign

4.3.4 Clustal W

Čas zarovnání: 0,0510 s

Score: 715

```
CLUSTAL 2.1 multiple sequence alignment


gi|377837259|ref|XM_003689219.      ATGGACGCTCAGAAAGATGTTCAACCACCAAAGCAGCAGCCAATGATATA 50
gi|262206272|ref|NR_029620.1|       ----GTGTTGGGGA-----CTCG--CGC----GCTGGGTCCAGTGGTTCT 35
gi|39980565|gb|AY377897.1|          CCCAGCGACGGGCAG------CGGCGGC---GGCGGCAGC-AGCAGCA-G 39
                                          *    * *       *     *    ** *   * *

gi|377837259|ref|XM_003689219.      TATTTGTGGAG-AGTGTCACACCGAAAATGAGATAAAGTCCAGGGATCCC 99
gi|262206272|ref|NR_029620.1|       TAACAGTTCAACAGTTCTGTAGCGCAATTGTGA-AATGTTTAGGA---CC 81
gi|39980565|gb|AY377897.1|          CAGCAGC--AGCAGCAGCA-ACAGCAGCCTCAGCAGCCGCAAGTG---CT 83
                                     *   *   *  **      *  * *        *      **     *

gi|377837259|ref|XM_003689219.      ATCAGATGCAGAGAATG--TAGATACAGAATAATGTACAAGAAAAGGACT 147
gi|262206272|ref|NR_029620.1|       ACTAGACCCGGCGGGCG--CGGCGACAG---------------------- 107
gi|39980565|gb|AY377897.1|          ATC---CTCGGAGGGCGGGCAGCTGCGGCA-------------------- 110
                                    *       * * *   *    *   * *

gi|377837259|ref|XM_003689219.      AAAAGATTGGTGGTTTTTGATGCTCGATGA 177
gi|262206272|ref|NR_029620.1|       ------------------------CGA--- 110
gi|39980565|gb|AY377897.1|          ------------------------CAAC-- 114
                                                           * *
```

Obrázek 14 - Vícenásobné zarovnání, sada dat 1 - ClustalW

4.3.5 T-Coffee

Čas zarovnání: 0,0435 s

Score: 37

```
T-COFFEE, Version_9.03.r1318 (2012-07-12 19:05:45 - Revision 1318 - Build 366)
Cedric Notredame
SCORE=37
*
 BAD AVG GOOD
*
gi|377837259|re    :  31
gi|39980565|gb|    :  36
gi|262206272|re    :  39
cons               :  37

gi|377837259|re       1 ATGGACGCTCAGAAAGATGTTCAACCACCAAAGCAGCAGCCAATGATATATATTTGTGGAGAG   63
gi|39980565|gb|       1 CCCAGCGA-CGGGCAGCG--GC------GGCGGCGGCAG-CAGCAGCAG-CAG-------CAG   45
gi|262206272|re       1 GTGT-----TGGGGACTC--GC-------G-CGCTGGGTCCAGTGGTTCTTAA-------CAG   41

cons                  1                                                               63


gi|377837259|re      64 TGTCACACCGAAAATGAGATAAAGTCCAGGGATCCCATCAGATGCAGAGAATGTAGATACAGA  126
gi|39980565|gb|      46 CAG--CA---G----CAGCA-----ACAGCAGCCTCAGCAGCCGCA---AGTGC---------   82
gi|262206272|re      42 TT---CA---A----CAGT------TCTGTAG-CGCAATTGTGAA-----ATGT---------   73

cons                 64                                                              126


gi|377837259|re     127 ATAATGTACAAGAAAAGGACTAAAAGA-TTGGTGGTTTTTGATGCTCGATGA  177
gi|39980565|gb|      83 ----TATCCTCG-----------GAGGGCGGGCA---GCTGCGGC--ACAAC  114
gi|262206272|re      74 -------TTAGG------ACCACTAGACCCGGCGGGCGCGGCGAC--AGCGA  110

cons                127                                                      178
```

Obrázek 15 - Vícenásobné zarovnání, sada dat 1 - T-Coffee

4.4 Výsledky zarovnání – Druhá sada dat

4.4.1 Implementovaný algoritmus pro tři sekvence

Čas zarovnání: 13,6057 s

Score: 435

```
AAATGCAACA-CTGCCACG-TGT-GCAACGCA-GCG--C-CTGGCAAATTTTT--T-AGTTCGTTCCA-GCA-A-C--A-AC-
--ATG--ACTTCTG--A-GAA-AGGAGAAGAAT-CGAACCCTGGG-A--GTTTGA-C-G-TC-TTCTATG-ACCCCAGAGA-A
TTCCCCCACCTTGGCCAGGAAGTGGCCCTGAATGCTAACACTAAGAACCAGAAGATCAGATGGAAAAATGAAGTCCGGATTCA


TT-T-GG----T----A-CCAT-T----CTCT-CATCTACCAA--CGTGGGAT-CC--GATA---C---AT-AT--
--CT---TCGT-AAAGAG----G-CCTG-TCTGC-TCTA-C--GA------A-A--TC-A-AGTG-GGG--CATGA
TTCTGGGTCTTTCCAGAGCCATGTCGAGCTTTCCAATGACCAAGAAAAGGCACACCTTGACATTGCAGGATCCTTA
```

Obrázek 16 - Vícenásobné zarovnání, sada dat 2 - implementovaný program

4.4.2 Implementovaný algoritmus pro progresivní zarovnání

Čas zarovnání: 0.68326 s

Score: 214

```
TTCCCCCACCTTGGCCAGGAAGTGGCCCTGAATGCTAACACTAAGAACCAGAAGATCAGATGGAAAAATGAAGTCCGGATT
ATGACTT-C-T-G---AGAAAGGAGAA--GAAT-CGAACCCTGGGAGTTTGACG-TCTTCT--A----TGACC-CCAGAGA
AAATGCAACACTG-CCAC---G------TG--TGC-AACGC-A-GCGCCTG--G--CAAATTTTTTAGTTCGTTCCAGCAA

CATTCTGGGTCTTTCCAGAGCCATGTCGAGCTTTCCAATGACCAAGAAAAGGCACACCTTGACATTGCAGGATCCTTA
-A--CT---TCGTAA-AGAGGCCTGTCT-GCT--C---T-AC---GAAAT--CA-AG-T-G-----G--GG--CATGA
CAA-CTTTG--GTACCATT-C--TCTC-ATCTA-CCAACGT----G----GG-AT-CC--GATA---CA---T-AT--
```

Obrázek 17 - Vícenásobné zarovnání, sada dat 2 - implementovaný program progresivní zarovnání

4.4.3 Matlab Multialign

Čas zarovnání: 0,0418 s

Score:

```
AAATGCAACA-CTGCCACG-TGT-GCAACGCA-GCG--C-CTGGCAAATTTTTT--T-AGTTCGTTCCA-GCA-A-C--A-AC-TT
--ATG--ACTTCTG--A-GAA-AGGAGAAGAAT-CGAACCCTGGG-A--GTTTGA-C-G-TC-TTCTATG-ACCCCAGAGA-A--
TTCCCCCACCTTGGCCAGGAAGTGGCCCTGAATGCTAACACTAAGAACCAGAAGATCAGATGGAAAAATGAAGTCCGGATTCATT


-T-GG----T----A-CCAT-T----CTCT-CATCTACCAA--CGTGGGAT-CC--GATA---C---AT-AT--
CT---TCGT-AAAGAG----G-CCTG-TCTGC-TCTA-C--GA------A-A--TC-A-AGTG-GGG--CATGA
CTGGGTCTTTCCAGAGCCATGTCGAGCTTTCCAATGACCAAGAAAAGGCACACCTTGACATTGCAGGATCCTTA
```

Obrázek 18 - Vícenásobné zarovnání, sada dat 2 -MATLAB multialign

4.4.4 Clustal W

Čas zarovnání: 0,01 s

Score: 660

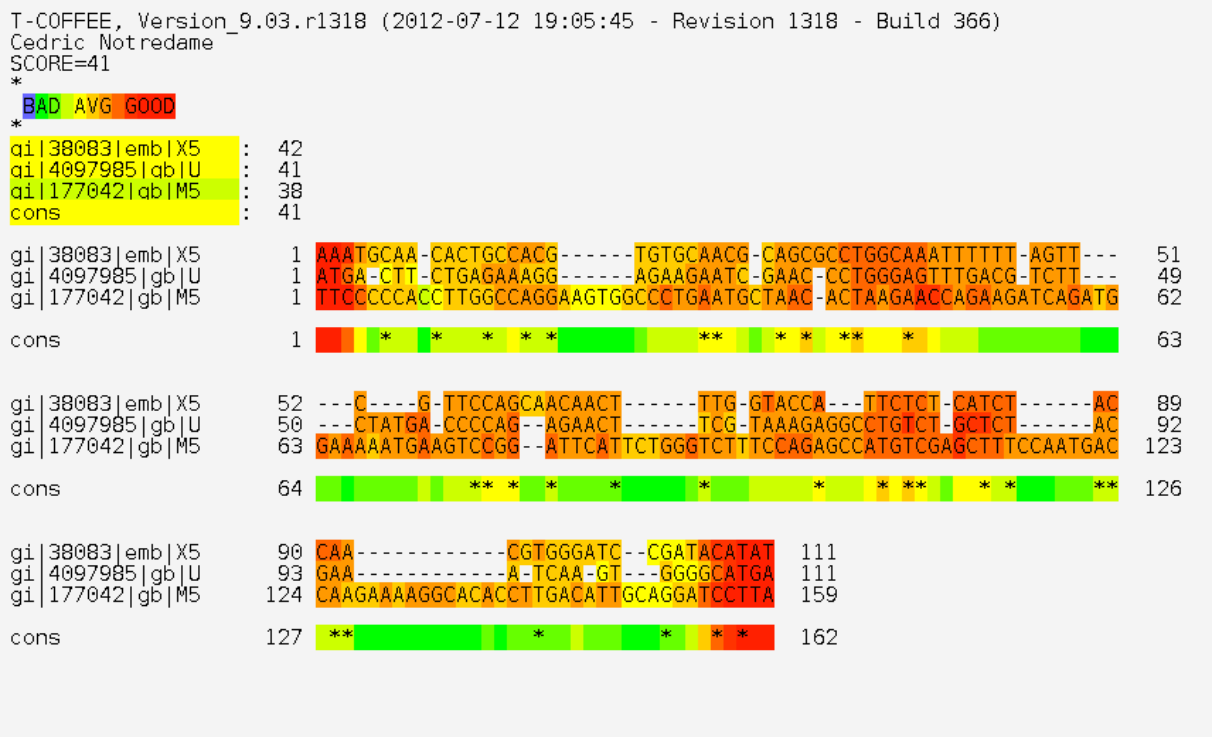
```
CLUSTAL 2.1 multiple sequence alignment


gi|38083|emb|X59792.1|              --------------------AAATGC-----AACACTGCCACGTGTGCAAC 26
gi|177042|gb|M59320.1|GORAPOB       TTCCCCCACCTTGGCCAGGAAGTGGCCCTGAATGCTAACAC-TAAGAACC 49
gi|4097985|gb|U72890.1|HSU7289      ----------------------ATG------ACTTCTGAGAAAGGAGAAGA 23
                                                 **       *   **   *     * *

gi|38083|emb|X59792.1|              GCAGCGCC---TGGCAAATT----TTTTAGTTCGTTCCAGCAACAACTTT 69
gi|177042|gb|M59320.1|GORAPOB       AGAAGATCAGATGGAAAAATGAAGTCCGGATTCATTCTGG--GTCTTTCC 97
gi|4097985|gb|U72890.1|HSU7289      ATCGAACC--CTGGGAGTTTGACGTCTTCTATGACCCCAGAGAACTTCGT 71
                                           *   *** *   *    *      *    *  *

gi|38083|emb|X59792.1|              G--GTACCAT------TCTCTCATCTACCAA------------CGTGGGA 99
gi|177042|gb|M59320.1|GORAPOB       A--GAGCCATGTCGAGCTTTCCAATGACCAAGAAAAGGCACACCTTGACA 145
gi|4097985|gb|U72890.1|HSU7289      AAAGAGGCCTGTCTGCTCTACGAA--ATCAAG-----------TGGGGCA 108
                                       *   * *        *   *   * ***              *  *

gi|38083|emb|X59792.1|              TCC---GATACATAT 111
gi|177042|gb|M59320.1|GORAPOB       TTGCAGGATCCTTA- 159
gi|4097985|gb|U72890.1|HSU7289      T-----GA------- 111
                                    *     **
```

Obrázek 19 - Vícenásobné zarovnání, sada dat 2 -ClustalW

4.4.5 T-Coffee

Čas zarovnání: 0,0319 s

Score: 41



Obrázek 20 - Vícenásobné zarovnání, sada dat 2 -T-Coffee

4.5 Výsledky zarovnání - Třetí sada dat

4.5.1 Implementovaný algoritmus pro progresivní zarovnání

Čas zarovnání: 5.8775 s

Score: 1592

Obrázek 21 - Vícenásobné zarovnání, sada dat 3, Algoritmus pro progresivní vícenásobné zarovnání

4.5.2 Matlab Multialign

Čas zarovnání : 0.18642 s

Score:

Obrázek 22 - Vícenásobné zarovnání, sada dat 3, Matlab Multialign

4.5.3 Clustal W

Čas zarovnání: 0.1

Score: 3108

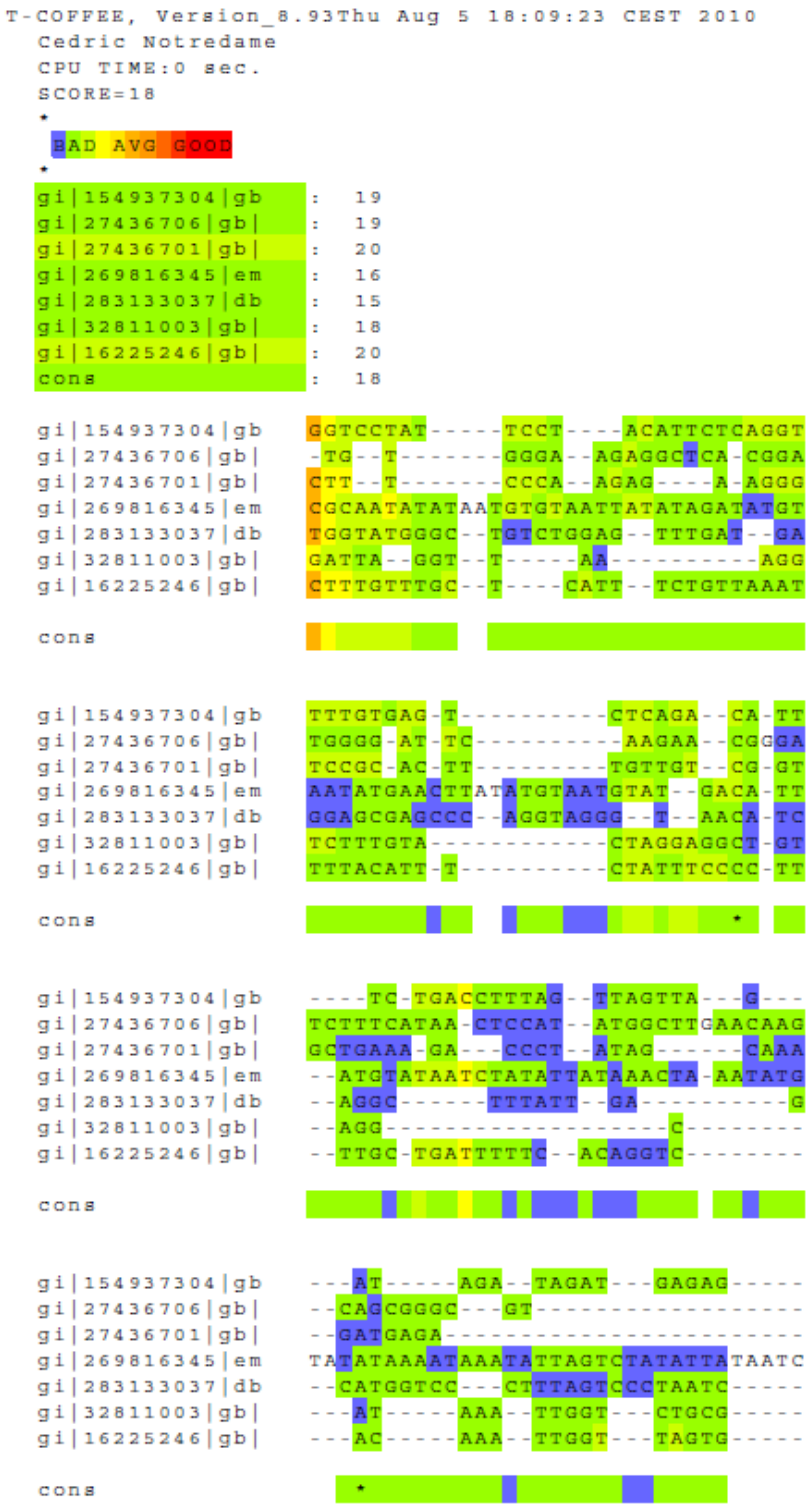
```
CLUSTAL 2.1 multiple sequence alignment


gi|154937304|gb|BV725426.1|         -------GGTCCTATTCCTAC--------------ATTC----------- 18
gi|16225246|gb|AF416919.1|AF41      ---------CTTTGTTTGCTC--------------ATTTC---------T 18
gi|27436706|gb|G73539.1|            ------------TGTGGGAAG--------------AGGC----------- 13
gi|27436701|gb|G73534.1|            -----------CTTTCCCAAG--------------AGA------------ 13
gi|283133037|dbj|AB473218.1|       --TGGTATGGGCTGTCTGGAGTTTGATGAGGAGCGAGCCC---------A 39
gi|269816345|emb|FN579877.1|       CGCAATATATAATGTGTAATT--------------ATATAGATATGTAAT 36
gi|32811003|gb|AF528225.1|         -----------------GATT--------------AGGT----------- 8
                                                     *

gi|154937304|gb|BV725426.1|         -TCAGGTT----TTGTGAGTCTCAGA---------------CATTTCTGA 48
gi|16225246|gb|AF416919.1|AF41      GTTAAATT----TTACATTTCT--------------------ATTTCC-- 42
gi|27436706|gb|G73539.1|            -TCACGGA----TGGGGATTCA--------------------AGAACGGG 38
gi|27436701|gb|G73534.1|            ---AGGGT----CCGCACTTT--------------------GTTGTCGG 35
gi|283133037|dbj|AB473218.1|       GGTAGGGTAACATCAGGCTTTATTG-----------------AGCATGGT 72
gi|269816345|emb|FN579877.1|       ATGAACTTA--TATGTAATGTATGACATTATGTATAATCTATATTATAAA 84
gi|32811003|gb|AF528225.1|         -TAAAGGTC--TTTGTACTAGGAGGC------------------TGTAGG 37
                                      *

gi|154937304|gb|BV725426.1|         CCT----------TTAGTTAGTT------AGAT---A------GATAGAT 73
gi|16225246|gb|AF416919.1|AF41      CCT----------TTTGCTGATTTTTCACAGGTCACA------AATTGGT 76
gi|27436706|gb|G73539.1|            ATC----------TTTCATAACTC-----------CA------TATGGCT 61
gi|27436701|gb|G73534.1|            TGC----------TGAAAGACCCT-------ATAGCA------AA--GAT 60
gi|283133037|dbj|AB473218.1|       CCC----------TTTAGTCCCTA-------ATCTTG------CCTTTGC 99
gi|269816345|emb|FN579877.1|       CTAAATATGTATATAAAATAAATATTAGTCTATATTATAATCTGATTTGT 134
gi|32811003|gb|AF528225.1|         C----------------ATAAAT--------------------TGGTCTGC 52


gi|154937304|gb|BV725426.1|         GAGAGAGA-------GAGAGAGAGAGAGAGAGAGA-GAGAGAGAGAGA 115
gi|16225246|gb|AF416919.1|AF41      TAGTGAAG-------TCATAGGAGCTTCTGACCTACT-TCTCTTGTTAG- 117
gi|27436706|gb|G73539.1|            TGAACAAG-------CAGCGGGCGTATACCACAGAGAAGCTCGGGCTGGC 104
gi|27436701|gb|G73534.1|            GAGAAGAC-------TTGTTCGGATTGAGCAGAGACAAAAAAGGCTCGAA 103
gi|283133037|dbj|AB473218.1|       CYGGGGAC-------CTATT----TTACCCA---ATTTTTAAGAGTAAGT 135
gi|269816345|emb|FN579877.1|       TCATGAACAAAGTTTCCCTTTAAGTTTTTCATTTTTTTTTTTTTTTTGAA 184
gi|32811003|gb|AF528225.1|         GCACCAGCA------CCATGCAACTTTTTCACCTCTGCCT---------A 87
                                                 *

gi|154937304|gb|BV725426.1|         GA----------------GAGAGAATCAA-------------CACTTTG- 135
gi|16225246|gb|AF416919.1|AF41      GT----------------GTGTAAACAGA-------------CATTTTG 138
gi|27436706|gb|G73539.1|            AG----------------ATACAAGCT---------------CACCTACG 123
gi|27436701|gb|G73534.1|            GG----------------AAATGAGTT----------------------- 114
gi|283133037|dbj|AB473218.1|       AT----------------AGA-AAGTA------------------TTGTG 150
gi|269816345|emb|FN579877.1|       ATGGAGTCTCACTCTGTTGCCCAGGCTGGAATGCAGTGGTACAACCTCCG 234
gi|32811003|gb|AF528225.1|         AT-------CA-TCTCTTGTTCATGTC-----------------CTACTG 112


gi|154937304|gb|BV725426.1|         -TGAACTTAA----------------TAAATACTTTCAGAAAAACCTGT 167
gi|16225246|gb|AF416919.1|AF41      CTGACCTTAACATG--------CCTTTTAAATGCTTCTA-ATAAACTAGT 179
gi|27436706|gb|G73539.1|            CCGAAGCCAAGGC------------CGTATGTGAATTTGAAGGTGGTCGT 161
gi|27436701|gb|G73534.1|            -TGAACGGGACA--------------GTGAGC-CATTCAGATGTCTCTGC 148
gi|283133037|dbj|AB473218.1|       TTGAA-GCAATAT--------------TGAGGAATCCTGAAAATGTTGAT 185
gi|269816345|emb|FN579877.1|       TTCACTGCAACATCTGCCTCCCAGGTTCAAGCAGTTCCC--TGCCTCAGC 282
gi|32811003|gb|AF528225.1|         TTCAA----------GCCTCC-------AAGCTGTGCCT--TGGGT-GGC 142
                                      *

gi|154937304|gb|BV725426.1|         GTGGTGGG---------- 175
gi|16225246|gb|AF416919.1|AF41      T----------------- 180
gi|27436706|gb|G73539.1|            CTCGCAACCTACAAGCA- 178
gi|27436701|gb|G73534.1|            A----------------- 149
gi|283133037|dbj|AB473218.1|       ACTCAAATTCTGATC--- 200
gi|269816345|emb|FN579877.1|       CTCCCAAGTAGCTGGGAT 300
gi|32811003|gb|AF528225.1|         TT----------TAGGGC 150
```

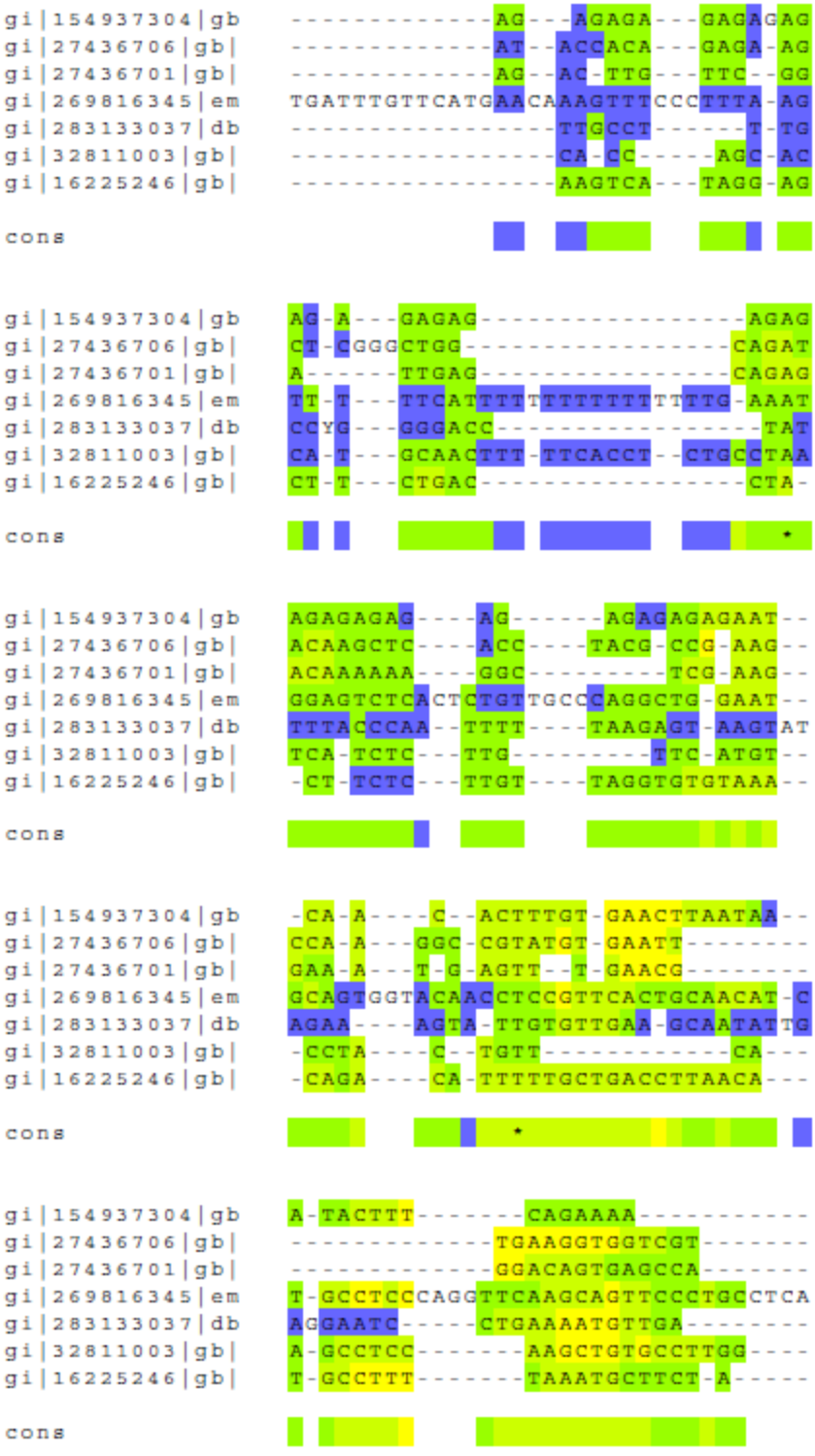
Obrázek 23 - Vícenásobné zarovnání, sada dat 3, ClustalW

4.5.4 T-Coffee

Čas zarovnání: 0 s

Score: 18




gi|154937304|gb
gi|27436706|gb|
gi|27436701|gb|
gi|269816345|em
gi|283133037|db
gi|32811003|gb|
gi|16225246|gb|
cons

Obrázek 24 - Vícenásobné zarovnání, sada dat 3, T-Coffee

5 Srovnání

Při zarovnávání sekvencí bylo celkem použito 5 programů, ačkoliv program pro zarovnání tří sekvencí se omezuje pouze na první dvě sady dat. Každý z programů postupuje při zarovnávání jinak, i když metody pro progresivní zarovnání jako je implementovaný program pro progresivní zarovnání a Clustal W sdílejí podobný přístup k problematice. Mimo řadu pak stojí T-Coffee, který pro iniciální zarovnání používá vstupy vytvořené algoritmem Clustal a na nich provádí vlastní analýzu. První tři metody, tedy implementovaný algoritmus pro zarovnání tří sekvencí, algoritmus pro progresivní zarovnání popsaný v této práci a integrovaná utilita pro vícenásobné zarovnání sekvencí programu MATLAB byly spouštěny lokálně na PC. Naopak druhé dva programy (Clustal W - Omega a T-Coffee) byly spouštěny jako server based aplikace. Což znamená, že pro jejich nasazení bylo využito výpočetních clusterů volně dostupných na internetu. Vzhledem k aktuálnosti použitých programů byl místo Clustal W použit program Clustal Omega, který představuje další krok ve vývoji přinášející vyšší přesnost při zarovnání sekvencí.

Prvním kritériem pro srovnání zarovnání bylo výsledné score. Bohužel pro defaultní funkci MATLAB Multiple sequence alignment neexistuje možnost, jak jej implicitně zobrazit. Druhým kritériem pak byl čas běhu programu, který je důležitým měřítkem pro jeho použitelnost v praxi, kdy je potřeba zarovnávat velké soubory sekvencí o mnoha znacích.

5.1 Score

Z výsledných score získaných vícenásobnými zarovnáními je zřejmé, že první sada sekvencí obsahuje méně si podobná data, ačkoliv ClustalW produkuje score vyšší než ve srovnání s druhou sadou sekvencí. Tento výkyv je možné připočíst rozdílnému způsobu sestavování zarovnání oproti ostatním metodám. Celkový výsledek potom ovlivňuje zařazení sekvence z myši domácí, která s ostatními lidskými sekvencemi nesdílí přílišnou podobnost, a tedy výrazně ovlivňuje celkový výsledek.

Druhá sada sekvencí oproti první obsahuje pouze sekvence primátů, které jsou si geneticky podobnější než předchozí sada. Tomu odpovídá jak vyšší score zarovnání, tak i celkově větší podíl konzervativních regionů, jak je zřejmé z výsledků ClustalW, T-Coffee a algoritmu pro progresivní zarovnání.

Třetí sada sekvencí zahrnuje různorodou kombinaci sekvencí, ale především z celkového počtu sedmi sekvencí tři patří myši domácí. Implementovaný algoritmus pro progresivní zarovnání sice vypočetl poměrně vysoké score, ale při bližším zkoumání je zřetelné, že první tři sekvence jsou si velmi podobné. Jedná se právě o sekvence mus muscullus, na kterých se projevuje jev popsaný v podkapitole věnované algoritmu Carillo-Lipman. Pokud se v balíku sekvencí vyskytne větší

počet evolučně si příbuzných sekvencí, tak může dojít k jevu, kdy tyto negativně ovlivní další zarovnání. Je jasně vidět, že zarovnání těchto sekvencí upozaďuje zbylé páry. Implementovaný algoritmus progresivního zarovnání však nepoužívá žádný mechanismus, který by tento nežádoucí jev potlačoval. Zde lze určitě spatřovat prostor pro budoucí vylepšení. Zbylé algoritmy si při zarovnání vedou lépe, především pro to, že v různých podobách zohledňují výskyt velmi podobných sekvencí a jejich vliv na celkové zarovnání. Pro poslední sadu sekvencí nebylo možné použít algoritmus pro zarovnání tří sekvencí, protože jeho činnost je omezena na pouhé tři sekvence.

5.2 Časová náročnost

Uvedené výsledné časy všech zarovnání poukazují na rozdílnost v náročnosti všech metod. Výrazně vyšší doba zarovnání je u implementovaného algoritmu v prostředí Matlab, což je dáno především opakujícími se cykly průchodu sestavenými 3D maticemi, jejichž údržba je náročná na operační paměť. Toto místo tak představuje prostor pro další optimalizaci zdrojového kódu. Naproti tomu druhý implementovaný algoritmus pro progresivní zarovnání naplno využívá své výhody. Není závislý na provádění výpočetních operací a hledání cesty skrze tří rozměrnou matici. Místo toho provádí rekurzivně, na základě guide tree, párové zarovnání, které je mnohem rychlejší a výpočetně jednodušší.

Jako nejslibnější v oblasti časové náročnosti se jeví programy ClustalW a T-Coffee. Náročnost, resp. nenáročnost vychází z jejich spouštění na systémech clusterů, které mají nesrovnatelně větší výkon oproti klasickým stolním počítačům. Druhým faktorem pro rychlost těchto metod zarovnání jsou použité algoritmy, které nevyžadují udržování náročných 3D matic pro průběh výpočtů zarovnání.

6 Závěr

Tato práce si kladla za úkol shromáždit a popsat dostupné algoritmy pro vícenásobné zarovnávání biologických sekvencí. Získané informace jsou rozčleněny do několika základních bloků, které hlouběji rozebírají danou problematiku. Nejprve byly stanoveny obecné principy a postupy, na nichž staví nejčastěji používané metody pro vícenásobné zarovnání sekvencí, včetně jejich postupu a metody určování kvality takto sestavených zarovnání. Další oddíl se zabývá popisem a implementací vybrané metody pro vícenásobné zarovnání v programovém prostředí MATLAB. Ze škály dostupných řešení pro vícenásobné zarovnání byla vybrána některá z nich, jmenovitě ClustalW, MUSCLE, Carillo-Lipman, Matlab Multiple sequence alignment a T-Coffee, která byla dále podrobněji popsána. Tři z takto popsaných algoritmů byly poté spolu s implementovanými algoritmy použity pro zarovnání vybraného souboru nukleotidových sekvencí. Za zmínku také stojí koncepce použitého algoritmu pro vícenásobné progresivní zarovnání, jehož objektové pojetí značně zjednodušilo implementaci, především díky znovupoužitelnosti jednotlivých komponent na různých místech kódu.

Použité sekvence byly získány z volně přístupné databáze NCBI. Výsledky pak byly zdokumentovány a doplněny o zobrazení výsledných zarovnání. Následně byly výsledky porovnány podle stanovených kritérií pro kvalitu zarovnání a celkovou náročnost použití implementovaných a zkoumaných algoritmů. Ze všech použitých algoritmů vykazuje právě Clustal W největší score zarovnání, které je schopen sestavit v relativně krátkém čase. Což z něj činí schopný nástroj pro porovnávání sekvencí. Popsané vlastnosti jsou výsledkem použitého přístupu algoritmu, na němž je Clustal W postaven a také to, že se jedná o jeden z nejstarších nástrojů pro vícenásobné zarovnání sekvencí. Právě dlouhá doba jeho vývoje umožnila velké množství vylepšení a optimalizací tohoto robustního nástroje. T-Coffee staví na výsledcích získaných předběžným zarovnáním Clustal W. Nicméně hodnota score je velmi nízká, což je způsobeno použitím odlišných skórovacích parametrů. Implementované algoritmy dosahují rovněž vysokých hodnot score. A i když oba zarovnávají sekvence rozdílným postupem, můžeme vysledovat například na první sadě sekvencí, kde oba zachovávají stejné konzervativní úseky. Ale v případě srovnání časové náročnosti je algoritmus progresivního zarovnání nesrovnatelně výkonnější. Samotné zarovnání je souborem relativně nenáročných výpočetních operací, které je tak možné provádět s minimálními prostředky. Navíc jeho koncepce rozdělení na menší výpočetní celky umožňuje upravovat způsob, jakým algoritmus zarovnání sestavuje.

Vícenásobné zarovnání nukleotidových sekvencí však nepředstavuje pouhý finální produkt porovnání souboru sekvencí, ale je také důležitým přípravným krokem pro fylogenetickou analýzu. Bez kvalitně provedené přípravy vstupních dat není myslitelné, že by bylo možné dosáhnout biologicky a evolučně přesných výsledků. Vzhledem k rozsáhlosti problematiky vícenásobného zarovnání a její aktuálnosti pro současný směr vývoje zkoumání biologických sekvencí je tak zcela jasné, že tato oblast bioinformatiky bude nabývat na stále větším významu.

7 Seznam použité literatury

[1]CHAO, Kun-Mao a Louxin ZHANG. *Sequence Comparison: Theory and Methods*. 2009: Springer-Verlag London Limited, London. ISBN 978-1-84800-319-4.

[2]PHILLIPS, Aloysius, Daniel JANIES a Ward WHEELER. Multiple sequence alignment in phylogenetic analysis. *Molecular Phylogenetics and Evolution* [online]. 2000, č. 16, 317–330 [cit. 2012-12-14]. Dostupné z: http://www.ncbi.nlm.nih.gov/pubmed/10991785

[3]OGDEN, T. HEATH a MICHAEL S. ROSENBERG. Multiple Sequence Alignment Accuracy and Phylogenetic Inference. *Syst. Biol.* [online]. 2006, č. 55, 314–328 [cit. 2012-12-14]. Dostupné z: http://www.ncbi.nlm.nih.gov/pubmed/16611602

[4]JONES, Neil C. a Pavel A. PEVZNER. An Introduction to Bioinformatics Algorithms. United States of America: MIT Press, 2004. ISBN 0-262-10106-8.

[5]THU HANG, Nguyen. COMPARISON OF MULTIPLE SEQUENCE ALIGNMENT PROGRAMS IN PRACTISE [online]. AARHUS, Oct, 2008 [cit. 2012-12-14]. Dostupné z: www.daimi.au.dk/~cstorm/students/Nguyen_Feb2009.pdf. Diplomová práce. University of Århus. Vedoucí práce Christian N. S. Pedersen.

[6]NOTREDAME, Cédric, Desmond G. HIGGINS a Jaap HERINGA. T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment. *T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment* [online]. 2000, č. 302, s. 205-217 [cit. 2012-12-14]. Dostupné z: http://www.ncbi.nlm.nih.gov/pubmed/10964570

[7]THOMPSON, Julie D., Toby J. GIBSON, Frédéric PLEWNIAK, François JEANMOUGIN a Desmond G. HIGGINS. The CLUSTAL_X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acids Research* [online]. 1997, roč. 25, č. 24, 4876–4882 [cit. 2012-12-14]. Dostupné z: http://www.ncbi.nlm.nih.gov/pubmed/9396791

[8]CHENNA, Ramu, Hideaki SUGAWARA, Tadashi KOIKE, Rodrigo LOPEZ, Toby J. GIBSON, Desmond G. HIGGINS a Julie D. THOMPSON. Multiple sequence alignment with the Clustal series of programs.*Nucleic Acids Research* [online]. 2003, č. 13, 3497–3500 [cit. 2012-12-14]. Dostupné z: http://www.ncbi.nlm.nih.gov/pubmed/12824352

[9]THOMPSON, Julie D., Desmond G. HIGGINS a Toby J. GIBSON. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research* [online]. 1994, roč. 22, č. 22, s. 4673-4680 [cit. 2012-12-14]. Dostupné z: http://www.ncbi.nlm.nih.gov/pubmed/7984417

[10]LARKIN, M.A., G. BLACKSHIELDS, N.P. BROWN, R. CHENNA, P.A. MCGETTIGAN, H. MCWILLIAM, F. VALENTIN, I.M. WALLACE, A. WILM, R. LOPEZ, J.D. THOMPSON, T.J. GIBSON a D.G. HIGGINS. Clustal W and Clustal X version 2.0. *BIOINFORMATICS APPLICATIONS NOTE* [online]. 2007, roč. 23, č. 21, 2947–2948 [cit. 2012-12-14]. Dostupné z: http://www.ncbi.nlm.nih.gov/pubmed/17846036

[11]LIPMAN, David J., Stephen F. ALTSHUL a John D. KECECIOGLU. A tool for multiple sequence alignment. *Proc. Nail. Acad. Sci. USA* [online]. 1989, č. 86, s. 4412-4415 [cit. 2012-12-14]. Dostupné z: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC287279/

[12]TRNĚNÝ, Ondřej. Pokročilé zarovnávání a určování genetické odlišnosti sekvencí DNA[online]. Brno, 2011 [cit. 2012-12-14]. Dostupné z: https://www.vutbr.cz/studium/zaverecne-prace?zp_id=39584. Bakalářská práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce Ing. MARTIN VALLA.

[13]CHEN, Shyi-Ming a Chung-Hui LIN. Multiple DNA Sequence Alignment Based on Genetic Simulated Annealing Techniques. *Information and Management Sciences* [online]. 2007, č. 18, s. 97-111 [cit. 2012-12-14]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.105.7718

[14]GUPTA, Sandeep K., John D. KECECIOGLU a Alejandro A. SCHAFFER. Improving the Practical Space and Time Efficiency of the Shortest-PAths Approach to Sum-of-Pairs Multiple Sequence Alignment. *J Comput Biol.* [online]. 1995, č. 21, s. 459-72 [cit. 2012-12-14]. Dostupné z: http://www.ncbi.nlm.nih.gov/pubmed/8521275

[15]VIJAN, Sonali a Rajesh MEHRA. Biological Sequence Alignment for Bioinformatics Applications Using MATLAB. *Int. J Comp Sci. Emerging Tech* [online]. 2011, č. 2, s. 310-315 [cit. 2012-12-14]. Dostupné z: http://ojs.excelingtech.co.uk/index.php/IJCSET/article/view/161

[16] SARKAR, Anoop. Multiple Sequence Alignment: A Brief Introduction. In: [online]. [cit. 2013-05-10]. Dostupné z: http://www.dynamics.unam.edu/DinamicaNoLineal2/investigacion/bioinformatica/msa.pdf

[17] MADĚRÁNKOVÁ, Denisa. Vícenásobné zarovnání. In: [online]. [cit. 2013-02-01]. Dostupné z: https://www.vutbr.cz/elearning/mod/resource/view.php?id=198282

[18] KOZAR, Lee. Multiple Alignment & Phylogenetic Analysis. In: [online]. s. 48 [cit. 2013-02-01]. Dostupné z: cmgm.stanford.edu/classes/pdf/phylogenetic.pdf

[19] LÖYTYNOJA, Ari a Nick GOLDMAN. An algorithm for progressive multiple alignment of sequences with insertions. Proceedings of the National Academy of Sciences of the United States of America [online]. Washington, D.C.: The Academy, 2005, roč. 30, č. 102, s. 6 [cit. 2013-05-02]. Dostupné z: http://www.pnas.org/content/102/30/10557.full.pdf+html

[20] EDGAR, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research [online]. 2004-03-08, vol. 32, issue 5, s. 1792-1797 [cit. 2013-05-01]. DOI: 10.1093/nar/gkh340. Dostupné z: http://www.nar.oupjournals.org/cgi/doi/10.1093/nar/gkh340

[21] EDGAR, Robert C. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics [online]. vol. 5, issue 1 [cit. 2013-05-20]. DOI: 10.1186/1471-2105-5-113. Dostupné z: http://www.biomedcentral.com/1471-2105/5/113

[22] MOEJBAEK. Exact Multiple Sequence Alignment using Forward Dynamic Programming. Bioinformatics Research Center Aarhus University, 2010. Dostupné z: http://cs.au.dk/~cstorm/students/Moejbaek_Mar2010.pdf. Diplomová práce. Aarhus University. Vedoucí práce Christian N. S. Pedersen.

[23] CVRČKOVÁ, Fatima. Úvod do praktické bioinformatiky. Vyd. 1. Praha: Academia, 2006, 148 s. ISBN 80-200-1360-1.

[24] RÉDEI, G. Encyclopedia of genetics, genomics, proteomics, and informatics. 3rd ed. New York: Springer, c2008, 1134 p. Springer reference. ISBN 978-140-2067-556.

[25] GRIFFITHS, Anthony J. An introduction to genetic analysis. 7th ed. New York: W.H. Freeman, c2000, xvii, 860 p. ISBN 07-167-3520-2

[26] MIZRACHI, I. The NCBI Handbook: GenBank: The Nucleotide Sequence Database [online]. Bethesda (MD): National Center for Biotechnology Information, 2002 [cit. 2013-05-19].

8 Seznam příloh

Příloha 1: Vývojové diagramy algoritmu pro vícenásobné progresivní zarovnání

Příloha 2: Pseudokódy jednotlivých metod algoritmu pro vícenásobné progresivní zarovnání

Příloha 3: Zdrojové kódy algoritmu pro vícenásobné progresivní zarovnání v programovacím jazyku MATLAB