Univerzita Komenského v Bratislave
Fakulta Matematiky, Fyziky a Informatiky

Použitie komparatívnej informácie pri hľadaní génov

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra informatiky FMFI
Školiteľ:	Mgr. Bronislava Brejová, PhD.
Evidenčné číslo:	7b45b16f-4496-405c-8c38-df349512dbe2

Bratislava, 2011 **Andrej Ridzik**

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Andrej Ridzik
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Použitie komparatívnej informácie pri hľadaní génov

Cieľ: Cieľom je implementovať modul na použitie celogenómových zarovnaní v systéme na hľadanie génov ExonHunter.

Vedúci: Mgr. Bronislava Brejová, PhD.
Katedra: FMFI.KI - Katedra informatiky

Dátum zadania: 19.10.2010

Dátum schválenia: 20.10.2010

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
Vedúci

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne s použitím citovaných zdrojov.

Ďakujem mojej školiteľke Mgr. Bronislave Brejovej, PhD.
za jej veľkú pomoc, cenné rady a trpezlivosť.

Abstrakt

Jedným zo základných problémov bioinformatiky je hľadanie (predikcia) génov v DNA sekvenciách. Tento problém je bežne riešený Viterbiho algoritmom na skrytých Markovových modeloch (HMM). Pri hľadaní génov sa často využívajú aj externé dáta - EST sekvencie, proteínové sekvencie, viacnásobné zarovnania DNA sekvencií a iné, ktoré môžu pri predikcii výrazne pomôcť.

Program ExonHunter okrem implementácie HMM obsahuje aj mechanizmus na spracovanie externých dát. Pôvodné implementácie nevyužívali možnosť využitia informácie z viacnásobných celogenómových zarovnaní, ktoré sme sa rozhodli do programu ExonHunter doplniť.

Upravený program s modulom, ktorý zabezpečuje využitie informácie zo zarovnaní, sme testovali na sade DNA sekvencií octovej mušky *Drosophila melanogaster*. Z dôvodu použitia jednoduchej skórovacej matice sa nám nepodarilo zaznamenať zlepšenie presnosti predikcie testovaných sekvencií. Predpokladáme však, že s použitím lepších matíc vytváraných známymi postupmi z literatúry dôjde k zlepšeniu presnosti a náš modul bude v budúcnosti slúžiť ako užitočná súčasť programu na hľadanie génov - ExonHunter.

Kľúčové slová: gén, zarovnanie, anotácia DNA, skrytý Markovov model, ExonHunter, externá informácia

Abstract

One of the essential challenges of bioinformatics is gene detection in DNA sequences. The problem is often solved by Viterbi algorithm for hidden Markov models (HMM). External data, e.g. EST sequences, protein sequences, multiple alignments of DNA sequences, etc., are also often used for improving predictions.

In addition to implementing the HMM, program ExonHunter contains a mechanism for processing external data. The original implementation does not allow the use of the information from multiple alignments, which we decided to add to the ExonHunter program.

The modified program with a module that implements the use of information from alignments, was tested on a set of DNA sequences from fruit fly *Drosophila melanogaster*. Due to simple scoring matrix, we were unable to observe improvement in prediction on the testing sequences. But we assume that using a matrix generated by known procedures from the literature will improve accuracy and our module will in future serve as a useful part of the program for gene finding - ExonHunter.

Keywords: gene, alignment, DNA annotation, hidden Markov model, ExonHunter, external evidence

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Posledné desaťročie prinieslo sprístupnenie genetických informácií mnohých organizmov a dalo podnet k rôznym novým otázkam. Jedným z dôležitých problémov súčasnej modernej biológie sa tak stal i problém hľadania génov. Ukázalo sa, že informatika môže významne pomôcť pri riešení otázok kde a aké gény sa u jednotlivých organizmov vyskytujú. Využívajú sa pritom rôzne metódy, ktoré okrem samotnej DNA sekvencie skúmaného organizmu môžu taktiež využívať aj tzv. externé dáta. Tieto dáta môžu pochádzať z rôznych zdrojov a môžu byť rôzneho typu, no ich základným významom je poskytnúť informáciu, ktorá môže pomôcť pri hľadaní génom. Jedným z programov na hľadanie génov, ktoré poskytujú možnosť využiť túto informáciu, je aj program ExonHunter.

Skúmanie evolučných zmien v podobných úsekoch genetických informácií príbuzných organizmov má potenciál poskytnúť užitočnú informáciu, ktorá sa následne môže využiť v programe na hľadanie génov [1]. Program ExonHunter touto schopnosťou doposiaľ nedisponoval a tak sme sa ju rozhodli doplniť.

V našej práci si najprv v prvej kapitole vysvetlíme niektoré pojmy z biológie, ktoré sú nutné k pochopeniu zvyšného textu. V druhej kapitole najprv uvedieme problém hľadania génov, následne vysvetlíme metódy ako sa v bioinformatike rieši a na záver uvedieme prístup, ktorý sme sa v našej práci rozhodli implementovať. Tretia kapitola už pozostáva z konkrétneho opisu nášho algoritmu a popisu formátov, s ktorými náš program pracuje. V poslednej kapitole sme opísali spôsob testovania úspešnosti výsledkov, analyzovali ich a navrhli možné zlepšenie.

Kapitola 1

Biologické pojmy

V nasledujúcej kapitole uvedieme niektoré základné pojmy z biológie, ktoré sú potrebné k lepšiemu pochopeniu ostatného textu. Za dôležité považujeme vysvetliť niekoľko termínov z oblasti molekulárnej biológie a genetiky. Čitateľ so základnou znalosťou v týchto oblastiach, môže nasledujúcu sekciu preskočiť.

1.1 DNA

DNA je skratka pre deoxyribonukleovú kyselinu (z angl. deoxyribonucleic acid). Je nositeľkou genetickej informácie bunky, ktorá riadi rast, delenie a regeneráciu bunky. DNA je tvorená polynukleotidovými vláknami a je väčšinou uložená v bunke v podobe dvojzávitnicovej špirály.

Základnou stavebnou jednotkou DNA sú *nukleotidy*. Nukleotid v DNA sa skladá z troch zložiek a to z fosfátovej zložky, sacharidovej zložky (doexyribóza) a purínovej alebo pyrimidínovej dusíkatej bázy. V nukleotidoch nachádzajúcich sa v DNA sa vyskytujú štyri druhy dusíkatých báz: *adenín, guanín, cytozín a tymín*. Jednotlivé nukleotidy sa od seba líšia len dusíkatými bázami a preto sa aj zvyknú označovať začiatočnými písmenami daných báz, teda *A, G, C a T*. Z tohto dôvodu v nasledujúcom texte nebudeme striktne rozlišovať pojmy báza a nukleotid.

Celková genetická informácia bunky sa nazýva *genóm* a je zakódovaná práve poradím nukleotidov A, G, C, T. V eukaryotických organizmoch[1] je väčšina genómu uložená v molekulách, ktoré sa nazývajú *chromozómy* a sú uložené v jadre bunky. Zvyšné malé časti genómu sa nachádzajú v mitochondriách a v prípade rastlín i v chloroplastoch. Ide o relatívne malé množstvo DNA molekúl, ktoré uskladňujú celkovú informáciu potrebnú k riadeniu života bunky. Napríklad u človeka to je len 46 molekúl (chromozómov).[3]

Jednotlivé konce polynukleotidového vlákna sa od seba navzájom odlišujú a preto ich zvykneme označovať *5' a 3'*. Dvojica vláken, ktoré spolu tvoria dvojzávitnicu, je medzi sebou prepojená vodíkovými väzbami medzi bázami, pričom sa spájajú dvojice

[1] Všetky živočíchy, rastliny, huby, ale aj niektoré jednobunkové organizmy

adenozín-tymín a guanín-cytozín. Hovoríme, že bázy A-T a G-C sú komplementárne. Jednotlivé vlákna sú teda k sebe opačne orientované a navzájom komplementárne. To znamená, že ak poznáme poradie báz jedného vlákna, tak ich komplementom a otočením celého reťazca dostaneme poradie báz na druhom vlákne dvojzávitnice.

Proces, pri ktorom sa z jednotlivých molekúl DNA získava genetická informácia (teda poradie dusíkatých báz) sa nazýva *sekvenovanie*. Na sekvenovanie DNA sekvencií bolo vymyslených už niekoľko technologických postupov, ktoré sa od seba líšia predovšetkým rýchlosťou a cenovou dostupnosťou. V súčastnosti sa podarilo úspešne osekvenovať už veľké množstvo DNA sekvencií rôznych organizmov, z ktorých sú mnohé prístupné i verejnosti prostredníctvom databázy GenBank [14].

1.2 Proteíny

Proteíny (alebo bielkoviny) sú nevyhnutnými zložkami všetkých rastlinných i živočíšnych buniek a plnia niekoľko funkcií. V podobe enzýmov sú nenahraditeľné pri regulácii rôznych biochemických reakcií, plnia stavebnú funkciu bunky, v podobe protilátok sa podieľajú na obranyschopnosti organizmu, môžu regulovať tvorbu ďalších proteínov a taktiež slúžia organizmu ako rezervné látky.

Proteíny sú tvorené reťazcom *aminokyselín*, ktorý môže byť rôzne dlhý. Poznáme 20 základných druhov aminokyselín a každej je priradené jedno písmeno abecedy (tab. 1.1). Proteíny, ako reťazce aminokyselín, vytvárajú vplyvom chemických väzieb medzi jednotlivými aminokyselinami rôzne zložité štruktúry. Ak sa daný proteín v organizme vyskytuje v podobe enzýmu, tak jeho funkcia veľmi často závisí práve od jeho tvaru. Ak majú teda dva rôzne proteíny veľmi podobný tvaru, tak môžu v organizme často plniť tú istú funkciu.

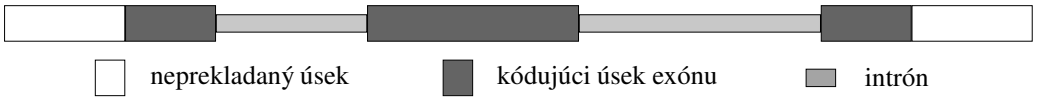
Tabuľka 1.1: Tabuľka aminokyselín a kodónov

Skratka		Aminokyselina	Kodóny
A	Ala	Alanín	GCA, GCC, GCG, GCT
C	Cys	Cysteín	TGC, TGT
D	Asp	Kys. asparágová	GAC, GAT
E	Glu	Kys. glutámová	GAA, GAG
F	Phe	Fenylalanín	TTC, TTT
G	Gly	Glycín	GGA, GGC, GGG, GGT
H	His	Histidín	CAC, CAT
I	Ile	Izoleucín	ATA, ATC, ATT
K	Lys	Lyzín	AAA, AAG
L	Leu	Leucín	CTA, CTC, CTG, CTT, TTA, TTG
M	Met	Metionín	ATG (Štart kodón)
N	Asn	Asparagín	AAC, AAT
P	Pro	Prolín	CCA, CCC, CCG, CCT
Q	Gln	Glutamín	CAA, CAG
R	Arg	Arginín	CGA, CGC, CGG, CGT, AGA, AGG
S	Ser	Serín	TCA, TCC, TCG, TCT, AGT, AGC
T	Thr	Treonín	ACA, ACC, ACG, ACT
V	Val	Valín	GTA, GTC, GTG, GTT
W	Trp	Tryptofán	TGG
Y	Tyr	Tyrozín	TAC, TAT
*	Stop		TAA, TAG, TGA (Stop kodóny)

1.3 Gén

Základnou jednotkou genetickej informácie je *gén*. Jedna z definícií hovorí, že je to časť DNA sekvencie, podľa ktorej sa zložitými biochemickými procesmi vytvorí jeden alebo viacero proteínov [3].

Gén pozostáva zo striedajúcich sa exónov a intrónov (obr. 1.1), pričom exóny tvoria úseky, ktoré kódujú proteíny, tj. reťazce aminokyselín. Na začiatku prvého a na konci posledného exónu v géne sa nachádzajú tzv. *neprekladané úseky* 5'UTR a 3'UTR (z angl. untranslated region), ktoré majú dôležitú úlohu pri procese vytvárania proteínu z génu, no žiadnu aminokyselinu nekódujú. Z tohto hľadiska teda možno celú DNA sekvenciu rozdeliť na úseky kódujúce aminokyseliny a na úseky nekódujúce a toto označenie budeme používať i v nasledujúcom texte.



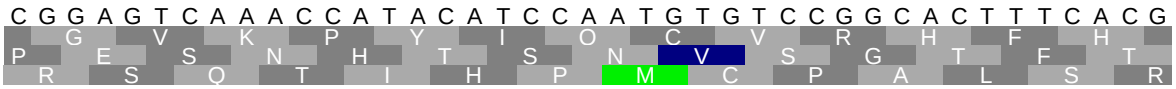
Obr. 1.1: Zjednodušená ukážka génu pozostávajúceho z neprekladaných úsekov, exónov a intrónov. Každá časť zodpovedá úseku DNA sekvencie s dĺžkou desiatok až tisícok báz

Každá aminokyselina je kódovaná trojicou nukleotidov, tzv. *kodónom* (tab. 1.1). Keďže počet rôznych trojíc nukleotidov je 64 a aminokyselín je len 20, tak niektoré aminokyseliny sú nutne kódované i viacerými kodónmi a preto aj proteín pozostávajúci z reťazca aminokyselín môže byť kódovaný viacerými postupnosťami kodónov.

Na začiatku kódujúcej oblasti prvého exónu sa vždy nachádza tzv. *štart kodón* (ATG). Špeciálnu úlohu tiež zohráva tzv. *stop kodón* (TAA, TAG, TGA), ktorý sa nachádza na konci kódujúcej oblasti posledného exónu a teda signalizuje ukončenie tvorby proteínu.

Preklad DNA sekvencie do proteínu je možný šiestimi rôznymi spôsobmi. Keďže sú kodóny trojperiodické, tak posun prekladu o jednu alebo dve bázy vedie k inému výsledku. Rozlišujeme teda tri *čítacie rámce*. Kódujúci úsek sa taktiež môže nachádzať na vlákne komplementárnom k tomu, na ktorom sa nachádza skúmaná sekvencia. Teda sekvencia môže byť prekladaná z dvoch možných vláken, v každom v troch možných čítacích rámcoch (obr. 1.2). Môžeme teda hovoriť o šiestich rôznych čítacích rámcoch a toto označenie budeme používať aj v ďalších častiach textu.

(a) Možnosti prekladu sekvencie v čítacích rámcov



(b) Možnosti prekladu sekvencie v čítacích rámcoch na opačnom vlákne

Obr. 1.2: Sekvencia CGTGAAAGTGCCGGACACATTGGATGTATGGTTTGACTCCG prekladaná do proteínu šiestimi možnými spôsobmi (obrázok vytvorený pomocou UCSC prehliadača génov[10])

Kapitola 2

Hľadanie génov

Jedným z hlavných biologickým problémov je problém hľadania génov. Ide o proces, pri ktorom sa v sekvenciách genómu organizmu hľadajú časti, ktoré predstavujú gén. Lokalizácia a popis funkcie génu tvorí základ pre pochopenie mnohých biologických procesov a taktiež môže pomôcť identifikovať príčinu rôznych genetických chorôb a následne ich skúmať. Pri riešení týchto problémov sa ukázala byť veľmi významným prostriedkom bioinformatika. Definujme si základný pojem sekvencia, s ktorým budeme v nasledujúcom texte často pracovať.

Pod pojmom *DNA sekvencia* sa v bioinformatike rozumieme slovo nad abecedou

$$\Sigma_D = \{A, G, C, T\},$$

teda každé slovo $w \in \Sigma_D^*$.
Podobne môžeme definovať pojem *proteín* ako *sekvenciu aminokyselín*, teda ako slovo nad abecedou

$$\Sigma_P = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$$

2.1 Problém anotácie DNA

Cieľom anotácie je priradiť jednotlivým úsekom DNA známe informácie o ich biologickej funkcii. Asi najdôležitejším využitím anotácie je práve hľadanie génov a následné popísanie ich funkcie.

Formálne môžeme anotáciu definovať nasledovne. Nech je G množina nejakých biologických pojmov. Pre problém hľadania génov si zvoľme napríklad jednoduchú množinu

$$G = \{\text{exón, intrón, medzigénový úsek}\}$$

Pre ľubovoľnú DNA sekvenciu $S, |S| = n$ definujeme jej anotáciu A_S ako prvok G^n. Teda každej báze sekvencie S priradíme nejaký biologický pojem, tj. označíme v akej

oblasti sa nachádza. Hľadať gény v DNA sekvencii S teda znamená priradiť jej takú anotáciu A_S, ktorá správne popisuje jej biologickú štruktúru.

Hľadanie génov prokaryotických organizmov je pomerne jednoduchá úloha, lebo ich genómy väčšinou obsahujú gény ako súvislé úseky, tj. exóny nie sú prerušované intrónmi. Stačí sa potom len pozerať na štart kodóny a stop kodóny a pomerne jednoduchými algoritmami danú oblasť anotovať. Na druhej strane hľadanie génov eukaryotických organizmov je zložitý problém, keďže ich genóm už môže mať komplikovanejšiu štruktúru [3].

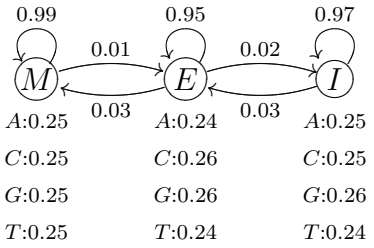
2.2 Skryté Markovove modely

Jedným zo základných spôsobov anotácie je použitie pravdepodobnostných modelov, tzv. *skrytých Markovových modelov* (hidden Markov models, HMM). Skrytý Markovov model H je pätica (K, Σ, π, a, e), kde K je konečná množina stavov, Σ je vstupná abeceda, π je pravdepodobnostná miera na množine stavov K, a je množina $a = \{a_q \mid q \in K\}$ pravdepodobnostných mier na množine K a e je množina $e = \{e_q \mid q \in K\}$ pravdepodobnostných mier na množine vstupnej abecedy Σ. Pre stav $q \in K$ je $\pi(q)$ pravdepodobnosť, že výpočet na HMM začne v stave q. Pre stavy $p, q \in K$ je $a_p(q)$ pravdepodobnosť, že HMM prejde zo stavu p do stavu q. Pre daný stav $q \in K$ a symbol $x \in \Sigma$ je $e_q(x)$ pravdepodobnosť, že HMM v stave q vygeneruje symbol x. Anotačná funkcia $f : K \to G$ transformuje postupnosť stavov výpočtu na HMM $Q = q_1 q_2 \dots q_n$ na postupnosť biologických pojmov $A = f(q_1)f(q_2)\dots f(q_n)$, čo tvorí samotnú anotáciu [7].

Pre daný model $H = (K, \Sigma, \pi, a, e)$ vieme sekvencii $S = s_1 s_2 \dots s_n$ a postupnosti stavov $Q_S = q_1 q_2 \dots q_n$ priradiť pravdepodobnosť, že HMM H vygeneruje S, pričom prejde cez stavy Q_S, ktorá je rovná

$$P(S, Q_S) = \pi(q_1) e_{q_1}(s_1) \prod_{i=2}^{n} a_{q_{i-1}}(q_i) e_{q_i}(s_i)$$

Na obrázku 2.1 je znázornený jednoduchý model na anotáciu DNA sekvencií, ktorý pozostáva z troch stavov a množín pravdepodobnostných mier a a e.



Obr. 2.1: Veľmi jednoduchý HMM na anotáciu DNA sekvencií. V skutočnosti sa používajú modely s oveľa zložitejšou štruktúrou. Stavy: M – medzigénový úsek, E – exón, I – intrón

Pri hľadaní anotácie DNA sekvencie S sa teda vytvorí skrytý Markovov model H, ktorý dobre modeluje biologickú realitu a následne sa nájde postupnosť stavov Q_S modelu H a výstupom bude anotácia A_S získaná anotačnou funkciou z Q_S. Na hľadanie týchto ciest sa používa Viterbiho algoritmus, ktorý je založený na dynamickom programovaní a bol vytvorený na dekódovanie tzv. konvolučných kódov [8].

2.3 Použitie externej informácie

Metóda hľadanie génov modelmi, ktoré využívajú len samotnú DNA sekvenciu sa označuje ako *ab-initio* predikcia. Keďže len pomocou samotnej genomickej DNA sekvencie nedokážeme spoľahlivo predikovať gény, väčšina programov na hľadanie génov ešte využíva *externé dáta*, ktorých úlohou je napomôcť k zvýšeniu presnosti predikcie. Tieto externé dáta môžu byť rôzneho druhu, napríklad môže ísť o databázu známych proteínov, o tzv. *EST sekvencie* (expressed sequence tags), tj. krátke sekvencie, ktoré sa často vyskytujú v exónoch alebo UTR úsekoch génu, DNA sekvencie iných príbuzných organizmov a mnohé iné. Jedinou nevýhodou oproti ab-initio predikcii je fakt, že pre skúmaný organizmus nemusíme práve požadovanými dátami disponovať, no ak ich k dispozícii máme, tak nám môžu pri hľadaní génov výrazne pomôcť.

2.3.1 ExonHunter

Jedným z programov určených na predikciu génov je program ExonHunter [5]. Svoju predikciu má založenú na využití skrytého Markovovho modelu spolu so spracovaním informácií z rôznych externých zdrojov. Jeho cieľom je skombinovať jednotlivé externé informácie, ktoré nazýva *radcami*, do výslednej jednej informácie, ktorú nazýva *super radcom*. Super radca výsledne ovplyvňuje modifikáciu Viterbiho algoritmu, ktorým sa rieši samotná anotácia [5]. Radca je definovaný nasledovne, pričom platí označenie z predchádzajúceho textu v časti 2.1:

Rada od radcu r pre bázu s_i pozostáva z rozkladu $\pi_{i,r}$ množiny G na triedy rozkladu a pravdepodobnostnej miery $p_{i,r}(t)$ na prvkoch tried rozkladu $t \in \pi_{i,r}$. Hodnota $p_{i,r}(t)$ je odhad pravdepodobnosti, že správna anotácia bázy s_i je prvkom triedy t, podľa informácií dostupných radcovi r.

Získané rady sa skombinujú do super radcu, pričom výsledná rada pre bázu s_i je čo najbližšie k radám od jednotlivých radcov [5].

Pre skúmanú sekvenciu S a anotáciu A nám HMM určuje pravdepodobnosť $P(A|S)$ s akou môže byť S anotovaná podľa A. Podobne nám super radca pre svoje informácie získané z externých dát E určuje pravdepodobnosť $P(A|E)$ s akou by mohli byť anotované anotáciou A. Naším cieľom je teda nájsť takú anotáciu A^*, pre ktorú bude pravdepodobnosť $P(A^*|S, E)$ maximálna. Super radca teda intuitívne zvyšuje pravdepodobnosť anotácii, ktorá je podporená externou informáciou. Po úpravách a zjednodušeniach sa následne na výpočet anotácie A^* aplikuje Viterbiho algoritmus [5][8].

2.4 Použitie zarovnaní pri hľadaní génov

Jedným z druhov externých dát používaných pri hľadaní génov sú aj zarovnania sekvencií príbuzných organizmov. Cieľom je porovnať našu vstupnú sekvenciu so známymi sekvenciami z databáz a nájsť medzi nimi podobné oblasti. Ako neskor ukážeme, štúdiom podobnosti a rozdielov medzi sekvenciami z príbuzných organizmov môžeme získať informáciu či skúmaná sekvencia môže obsahovať alebo byť súčasťou nejakého génu kódujúceho proteín. Podobne môžeme zarovnávať sekvencie aminokyselín, no v nasledujúcom texte sa budeme zaoberať len zarovnaním DNA sekvencií.

2.4.1 Lokálne a globálne zarovnania sekvencií

V bioinformatike pod pojmom *zarovnanie* sekvencií rozumieme vloženie medzier na niektorých miestach v jednotlivých sekvenciách za účelom spárovania podobných úsekov. V zarovnaní sa teda párujú bázy jednej sekvencie s bázami druhej sekvencie (obr. 2.2). Ak sa v zarovnaní spárovali dve rôzne bázy, tak hovoríme, že prišlo k mutácii danej bázy. Miesta, na ktorých bolo potrebné vložiť medzery, reprezentujú oblasti, v ktorých prišlo k vymazaniu niekoľkých báz. Hovoríme, že v sekvencii prišlo k *delécii*, resp. že v druhej sekvencii, ku ktorej sa zarovnáva, prišlo k *inzercii*.

Na ohodnotenie zarovnaní sa používajú *skórovacie schémy*, ktoré popisujú ich kvalitu. Jednoduchá skórovacia schéma môže ohodnocovať spárovanie rovnakých báz hodnotou +1, spárovanie rozdielnych báz -1 a výskyt inzercie, resp. delécie tiež hodnotou -1. Pomocou takejto schémy už vieme zarovnania porovnávať a hľadať zarovnanie s najväčším skóre.

Rozlišujeme dva druhy zarovnania a to lokálne a globálne zarovnanie (obr. 2.2). Pri globálnom zarovnaní sa snažíme čo najlepšie spárovať celé sekvencie. Pri lokálnom zarovnávaní hľadáme najlepšie možné spárovanie len časti prvej a druhej sekvencie. Môže teda nastať situácia, že najlepšie globálne zarovnanie bude zároveň aj najlepším lokálnym, no vo všeobecnosti to neplatí.

```
CA-GTCCTAGA
||  |||  |   |
CATGTCAT--A
```
(a) Globálne zarovnanie

```
CA-GTCCTA
||  |||  ||
CATGTCATA
```
(b) Lokálne zarovnanie

Obr. 2.2: Globálne a lokálne zarovnanie sekvencie CAGTCCTAGA a sekvencie CATGTCATA. Vertikálne čiary predstavujú spárovanie rovnakých báz sekvencie, znak pomlčky „-“ inzerciu, resp. deléciu

Na hľadanie globálneho a lokálneho zarovnania s najvyšším skóre pre danú schému existuje viacero algoritmov, ktoré sú väčšinou založené na dynamickom programovaní [3].

2.4.2 Viacnásobné celogenómové zarovnania

V našej práci sme pracovali so špeciálnym typom zarovnaní nazývaným *viacnásobné celogenómové zarovnanie*.

Pod slovom *celogenómové* sa rozumie, že zarovnávanými sekvenciami sú celé genómy organizmov. Cieľom zarovnania je porovnať podobné úseky, no ak by sme celé genómy zarovnávali klasickým spôsobom opísaným v časti 2.4.1, tak by sme vo všeobecnosti nedosiahli požadovaný výsledok, lebo jednotlivé podobné úseky sa nemusia v jednotlivých genómoch vyskytovať v tom istom poradí. Preto treba pri vytváraní celogenómových zarovnaní postupovať odlišne.

Jeden z organizmov, ku ktorých genómom sa zarovnanie vytvára, sa označí ako *referenčný organizmus* a druhý ako *informant*. Cieľom je zarovnať jednotlivé časti genómu informanta k referenčnému genómu. Vytvorenie najlepšieho celogenómového zarovnania sa väčšinou realizuje tak, že sa najprv hľadajú lokálne zarovnania a tie sa následne filtrujú a spájajú do dlhších úsekov [3].

Ak máme k dispozícii viacero príbuzných sekvencií, ktoré by sme chceli navzájom porovnať, tak môžeme vytvoriť zarovnanie týchto všetkých sekvencií (obr. 2.3). Tomuto typu zarovnania viacerých sekvencií hovoríme *viacnásobné zarovnanie*.

```
sekvencia 1: ATGCTG---ATAGGG
sekvencia 2: ATGCTCAAGATAGGG
sekvencia 3: ATGCTCAAA---GGG
sekvencia 4: ATCCT--AAATGGGC
```

Obr. 2.3: Viacnásobné globálne zarovnanie štyroch DNA sekvencií

Kombináciou týchto dvoch typov zarovnaní je práve viacnásobné celogenómové zarovnanie. K celému genómu referenčného organizmu sa teda získavajú zarovnania viacerých informantov. Dostaneme tak zarovnania jednotlivých podobných úsekov a keďže zarovnávame genómy viacerých príbuzných organizmov, tak nám zároveň podávajú lepšiu informáciu o zarovnaných častiach. Na vytvorenie viacnásobných celogenómových zarovnaní sa taktiež používa viacero algoritmov [3]. Zarovnania, s ktorými sme pracovali my v našej práci, boli vytvorené programom *Multiz* [9].

2.4.3 Skórovanie zarovnaní pri hľadaní génov

V tejto práci nás bude predovšetkým zaujímať, ako sa dá informácia získaná z celogenómových zarovnaní využiť pri probléme hľadania génov. Ak je DNA sekvencia referenčného organizmu génom, ktorý kóduje nejaký proteín, tak s veľkou pravdepodobnosťou budú daný gén obsahovať aj príbuzný informant. V kódujúcich častiach DNA sekvencií preto dochádza k mutáciám v menšom množstve, alebo dochádza k mutáciám kodónov, pri ktorých sa nemení kódujúca aminokyselina lebo to neovplyvní výsledný proteín, ktorého tvorba je v géne zakódovaná. Na druhej strane v nekódujúcich oblastiach DNA sekvencií často dochádza k mutáciám, ktoré by mohli zásadným spôsobom meniť štruktúru potenciálneho kódovaného proteínu. Viacnásobné celogenómové zarovnania nám poskytujú informácie o predpokladaných mutáciách a preto môžu pri predikcii génov slúžiť ako externé dáta.

Lin a kol.[1] zistili, že pozorovanie frekvencie substitúcií rôznych kodónov (*Codon substitution frequency*) v kódujúcich a nekódujúcich oblastiach môže zvýšiť presnosť predikcie génov. Svoj prístup použili na DNA sekvencie mušky *Drosophila melanogaster* a viacnásobné celogenómové zarovnania ďalších dvanástich príbuzných druhov hmyzu.

Pomocou pozorovaní známych kódujúcich a nekódujúcich oblastí trénovacích sekvencií si vytvorili pre substitúcie kodónov *skórovaciu tabuľku*. Hodnota skóre pre substitúciu potenciálnych kodónov hovorí ako často sa táto substitúcia vyskytuje v kódujúcich oblastiach v porovnaní s nekódujúcimi. K výpočtu jednotlivých hodnôt skórovacej tabuľky si natrénovali dve pomocné tabuľky substitúcií CSM^C a CSM^N [2]. Hodnota $\mathrm{CSM}_{A,B}$ predstavuje pravdepodobnosť s akou sa kodón B vyskytuje v sekvencii informanta, ak bol zarovnaný s kodónom A referenčného organizmu a zároveň $A \neq B$. Teda

$$\mathrm{CSM}_{A,B} = P(\text{kodón informanta}B \mid \text{kodón ref. organizmu}A, A \neq B)$$

Na kódujúcich a zvlášť na nekódujúcich častiach trénovacích sekvencií spočítali jednotlivé pravdepodobnosti zarovnania kodónov A a B, ktoré predstavujú jednotlivé hodnoty tabuliek CSM^C (pre kódujúce oblasti) a CSM^N (pre nekódujúce oblasti). Výslednú hodnotu skórovacej tabuľky pre kodóny A a B následne vyrátali ako

$$\log \frac{CSM^C_{A,B}}{CSM^N_{A,B}}$$

Skóre pre kodóny A a B je teda kladné, ak sa substitúcia A,B vyskytovala častejšie v kódujúcich úsekoch a záporné, ak v nekódujúcich úsekoch. Týmto spôsobom vytvorili pre každého informanta samostatnú skórovaciu tabuľku substitúcií kodónov.

Jednotlivým potenciálnym kodónom skúmanej sekvencie (i na komplementárnom vlákne a vo všetkých čítacích rámcoch) potom priradili finálne skóre, ktoré predstavovalo medián skóre substitúcií pre jednotlivé druhy hmyzu [2]. Tieto informácie výsledne použili ako externú informáciu pre program na hľadanie génov, pričom zaznamenali vysokú úspešnosť pri predpovedaní už známych génov. Taktiež navrhli niekoľko opráv v referenčnej anotácii na základe nových predikcií svojho programu a ich experimentálnej verifikácie.

Kapitola 3

Implementácia programu

Cieľom tejto práce je implementovať do programu ExonHunter nového radcu, ktorý bude spracúvať informáciu z celogenómových zarovnaní a jeho rady budú pomáhať programu ExonHunter pri predikcii. Ide o prídavný modul, ktorý rozhodne, ktoré časti skúmanej sekvencie môžu byť kódujúce alebo nekódujúce. Pri implementácii sme sa rozhodli inšpirovať myšlienkou z článku Lin a kol.[1], ktorú sme opísali v predchádzajúcej časti textu.

Súčasné časti programu ExonHunter, ktoré predspracovávajú externé informácie, sú implementované v programovacom jazyku *Perl* a preto sme sa ho rozhodli využili i pri tvorbe nášho nového radcu *mafcsf*. Modifikovali sme existujúci modul *program-wrapper* programu ExonHunter, ktorý má za úlohu spúšťať externé programy a modifikovať ich predikcie do GTF formátu. My sme do tohto modulu pridali možnosť vytvoriť externú informáciu z celogenómových zarovnaní. Hlavná časť výpočtu je vykonávaná v samostatnom Perl skripte *compute_ mafcsf.pl*.

Náš program k svojej činnosti potrebuje niekoľko vstupných súborov: sekvencie určené na anotáciu, viacnásobné zarovnanie pokrývajúce anotované sekvencie (v našom prípade celogenómové zarovnanie), skórovacia tabuľka a konfiguračný súbor s nastaveniami.

Náš program najprv ku konkrétnej sekvencii vyextrahuje príslušnú časť zarovnania a následne jednotlivým potenciálnym kodónom vo všetkých čítacích rámcoch priradí podľa skórovacej tabuľky určité skóre. Podľa hodnoty tohto skóre sa jednotlivé kodóny prehlásia za kódujúce alebo nekódujúce a táto informácia bude v podobe výstupného súboru podaná programu ExonHunter, ktorý ju spolu s externými informáciami od ostatných radcov využije pri jeho predikcii.

3.1 Formáty súborov

Program pracuje s viacerými typmi súborov. V nasledujúcej časti popíšeme formáty vstupných súborov nášho programu i formát jeho výstupu.

3.1.1 FASTA

FASTA formát sa používa na uchovávanie DNA a RNA sekvencií, alebo sekvencií aminokyselín vytvárajúcich bielkoviny. V jednom FASTA súbore môžu byť uložené viaceré sekvencie, pričom každá začína jednoriadkovou hlavičkou. Tento popisný riadok môže byť rôzneho tvaru a je odlíšený od zvyšných riadkov začínajúcim znakom „>“. Odporúča sa, aby žiaden riadok nepresahoval dĺžku 80 znakov. Príklad sekvencie vo FASTA formáte uvádzame na obrázku 3.1.

```
>dm3:chr2L:1996774:+:-1
TTGGGAACGTGCTTGGCATTGCCAATGAGCCTTCATTTTGTCCGAGGCTTTGTCCTTGTC
CCAGGATGTTTATGGACTGAAAGCAATGCGAGATTTGCGATAATGAGTATGATAAACATT
TTTATGGCGGCAGGTGACAAACAAAGATAACCATCGGAAGGCATCTTTAAGGGAGAGGAA
GCCAGGAAATTATTCAACAAATGGGTGGCGGCTTAATTAATTTAGAGCTCTACTAATGTA
ATTAGAAAAGAAGATAGTCTAAGTACCAATATGCATACTT
```

Obr. 3.1: Sekvencia vo FASTA formáte

V našom programe pracujeme len s DNA sekvenciami a rozlišujeme päť druhov znakov:

A – adenozín
C – cytozín
G – guanín
T – tymín
N – jedna z báz A,C,G,T (neznáma báza)

FASTA súbory používame v našom programe na uloženie vstupných sekvencií, ku ktorým sa majú vytvoriť predikcie kódujúcich a nekódujúcich oblastí.

Hlavička sekvencie

Pre hlavičky jednotlivých sekvencií vo FASTA súbore sme sa rozhodli použiť formát ako v programe TBA (*Threaded-Blockset Aligner*)[9]. Hlavička má tvar:

>string1:string2:int1:char:int2

- string1 – identifikácia organizmu, obvykle meno druhu
- string2 – názov kontigu (časti genómu), v našom prípade chromozómu
- int1 – pozícia začiatku sekvencie vzhľadom na kontig, pozície sa číslujú od 1
- char – rozlišuje, z ktorého vlákna sekvencia pochádza („+“ alebo „-“)
- int2 – dĺžka celého kontigu alebo „-1“ pre nedefinovanú hodnotu

Tento formát nám umožňuje mať vo FASTA súbore len krátky úsek určitého chromozómu a pritom mať poznamenané, z ktorého miesta na chromozóme tento úsek pochádza. Táto informácia bude dôležitá pri získavaní zarovnaní informantov k danému úseku.

3.1.2 MAF

V našom programe pracujeme s celogenómovými zarovnaniami. Teda chceme k jednému referenčnému genómu získať zarovnania genómov viacerých informantov. K tomuto účelu sme si zvolili formát MAF (*Multiple Alignment Format*). Je to formát pre viacnásobné zarovnania sekvencií, ktorý zabezpečuje ukladať niekoľko zarovnaní pre viacerých informantov v tvare, ktorý je pomerne ľahko spracovateľný i čitateľný. Z toho dôvodu ho je možné použiť i na ukladanie celogenómových zarovnaní.

Súbor je rozdelený do blokov, ktoré sú od seba oddelené prázdnym riadkom. Každý blok predstavuje zarovnanie pre určitý úsek referenčného genómu. Dĺžka týchto zarovnaní môže byť ľubovolná. V našom prípade celogenómových zarovnaní je teda celogenómové zarovnanie tvorené zarovnaniami v jednotlivých blokoch. S takýmito zarovnaniami sme pracovali aj v súboroch, ktoré sme používali, pričom boli bloky usporiadané podľa začiatku v referenčnej sekvencii.

Každé viacnásobné zarovnanie sa nachádza v samostatnom bloku, ktorý začína „a“-riadkom (riadok začínajúci znakom „a“) a obsahuje „s“-riadok (riadok začínajúci znakom „s“) pre každú sekvenciu vo viacnásobnom zarovnaní. Prvý „s“-riadok v bloku predstavuje sekvenciu referenčného organizmu, zvyšné sekvencie jednotlivých informantov. Niektoré druhy MAF súborov obsahujú aj iné typy riadkov, no tie náš program k predikcii nepotrebuje a preto sú ignorované. Ukážka jedného bloku MAF súboru sa nachádza na obrázku 3.2.

„a“-riadok slúži na identifikáciu začiatku nového bloku zarovnania. Taktiež sa tu môže nachádzať informácia o hodnotení zarovnania v tomto bloku, ktoré mu bolo pridelené jeho tvorcom.

„s“-riadky spolu s „a“-riadkom definujú viacnásobné zarovnanie. „s“-riadok pozostáva z niekoľkých zložiek, ktoré sú od seba oddelené bielymi znakmi. Riadok je v tvare:

⟨src⟩ ⟨start⟩ ⟨size⟩ ⟨strand⟩ ⟨srcSize⟩ ⟨text⟩

- ⟨src⟩
  Názov jednej zo zdrojových sekvencií v zarovnaní. Často sa používa tvar „organizmus.chromozóm“.

- ⟨start⟩
  Začiatok zarovnaného úseku zdrojovej sekvencie. Čísluje sa od 0.

- ⟨size⟩
  Veľkosť zarovnanej oblasti. Určuje počet báz v položke ⟨text⟩.

- ⟨strand⟩
  Určuje, ktoré z vláken („+“ alebo „-“) bolo zarovnané. Táto informácia je pre našu predikciu nepodstatná.

- ⟨srcSize⟩
  Celková veľkosť zdrojovej sekvencie, nie len veľkosť zarovnanej oblasti.

- ⟨text⟩
  Nukleotidy (alebo aminokyseliny) v zarovnaní spolu s prípadnými inzerciami (znak pomlčky „-“).

```
a score=14268.000000
s dm3.chr2L                 2420 33 +    23011544   CAACCCAAAATGGTGGCGGATGAACGAGATGAT
s droSim1.chr3R              427 33 -    27517382   caacccaaaatggcggcggacgaacgagatgac
i droSim1.chr3R             C 0 I 5515
s droSec1.super_14          1331 23 +     2068291   ccacacataatggcggcgtacga----------
i droSec1.super_14          C 0 I 26
s droYak2.chr3R             2895 33 -    28832112   CAATTCAAAATGGAGGTGATCGAAAGAGAGGGA
i droYak2.chr3R             I 1 C 0
s droWil1.scaffold_189243    390 33 +        1422   caacccaaaatggtggcggatgaacgagatgat
i droWil1.scaffold_189243   C 0 C 0
```

Obr. 3.2: Blok viacnásobného zarovnania vo formáte MAF

3.1.3 GTF

GTF je skratka pre *Gene Transfer Format*. Vznikol na základe formátu GFF (*General Feature Format*) a používa sa na anotáciu sekvencií. Anotácia, ktorá bola vytvorená naším programom, bola zapísaná práve v GTF formáte. Taktiež i celkové predikcie programu ExonHunter (viď. sekcia 2.3.1) sa zapisujú v tomto formáte.

Každý riadok v GTF formáte predstavuje jeden záznam pre časť sekvencie a pozostáva z ôsmich povinných položiek a z nepovinného zoznamu atribútov a poznámok. Štruktúra jedného riadku v GTF formáte je:

⟨seqname⟩ ⟨source⟩ ⟨feature⟩ ⟨start⟩ ⟨end⟩ ⟨score⟩ ⟨strand⟩ ⟨frame⟩ [attributes] [comments]

Jednotlivé položky sú oddelené jedným tabulátorom a žiadnymi bielymi znakmi. Položky sú definované nasledovne:

- ⟨seqname⟩
  Meno sekvencie. Zvyčajne je to ID chromozómu alebo ID časti DNA. V našom programe sme za meno sekvencie považovali jej popis v TBA formáte, ktorý sme popísali v sekcii 3.1.1

- ⟨source⟩
  Položka zdroja, obsahuje názov programu alebo verejnej databázy, odkiaľ daná anotácia pochádza. V našom programe na tomto mieste uvádzame *mafcsf*

- ⟨feature⟩
  Typ záznamu. Očakávané sú typy ako napr. „CDS“, „start_codon“, „stop_codon“, „exon“. Náš program vypisuje dva typy záznamov:
  - „CDS“ pre kódujúce úseky
  - „noncoding“ pre nekódujúce úseky
- ⟨start⟩ ⟨end⟩
  Celé číslo udávajúce relatívny začiatok a koniec danej časti sekvencie vzhľadom na začiatok sekvencie pomenovanej v ⟨seqname⟩, ⟨start⟩ musí byť menšie alebo rovné ⟨end⟩. Čísluje sa od 1.
- ⟨score⟩
  Skóre predstavuje stupeň spoľahlivosti s akou možno úsek považovať za typ ⟨feature⟩. Všeobecne platí: Čím vyššia hodnota, tým skôr je daný úsek ⟨feature⟩.
- ⟨strand⟩
  Môže nadobúdať hodnoty „+“, „-“ alebo znak bodky „.“ ak je hodnota neznáma alebo je táto informácia nepodstatná. Určuje, či sa anotovaná oblasť nachádza na danom vlákne (možnosť „+“) alebo na opačnom komplementárnom vlákne (možnosť „-“). Ak je teda vlákno „-“, tak prvá báza regiónu je komplementárna báza k báze na pozícii ⟨end⟩, keďže anotovaná oblasť začína na pozícii ⟨end⟩ a končí na ⟨start⟩ na opačnom vlákne.
- ⟨frame⟩
  0 znamená, že anotovaná oblasť začína celým kodónom na mieste prvej bázy oblasti. 1 znamená, že pred prvým anotovaným kodónom sa nachádza navyše jedna báza[1], tj. druhá báza regiónu predstavuje prvú bázu kodónu a 2 znamená, že tretia báza regiónu je prvou bázou kodónu anotovanej oblasti.
- [attributes]
  Predstavuje rôzne atribúty anotovanej oblasti. V našom programe túto položku vynechávame.

3.1.4 Skórovacia tabuľka

Náš program pri skórovaní používa skórovaciu tabuľku, ktorú načíta z jednoduchého textového súboru. Tabuľka udáva aké skóre má byť pridelené substitúcii kodónu A na kodón B.

Každý riadok súboru pozostáva z troch stĺpcov. V prvom stĺpci sa nachádza kodón A, v druhom stĺpci kodón B a v treťom sa definuje skóre pre danú substitúciu. Ak

[1] Tretia báza predchádzajúceho kodónu

je substitúcia považovaná za substitúciu v kódujúcej oblasti, tak je skóre kladné. Ak je substitúcia typickejšia pre nekódujúce oblasti, tak sa jej v tabuľke priradí skóre záporné.

3.1.5 Konfiguračný súbor

V konfiguračnom súbore sú uchované nastavenia pre náš program. Údaje sú uložené v štandardnom Unix-ovom konfiguračnom formáte, tj. obsahuje riadky
PREMENNÁ = HODNOTA a riadky s komentárom začínajúce úvodným znakom „#“. Na obrázku 3.3 vidíme ukážku konfiguračného súboru. V súbore sú nastavené hodnoty uvedených premenných:

- WINDOW – veľkosť okna
  Zvolená veľkosť okna, ktorému bude prideľované skóre. Jeho veľkosť musí byť v tvare $3k + 2$, aby bol počet potenciálnych kodónov vo všetkých čítacích rámcoch rovnaký (teda rovný k). Jeho presné využitie je vysvetlené v sekcii 3.2.3

- MIN_SCORE – dolná hranica skóre okna pre kódujúcu oblasť
  Kladné číslo, okná s aspoň takouto hodnotou sa budú považovať za kódujúce okná

- MAX_SCORE – dolná hranica skóre okna pre nekódujúcu oblasť
  Záporné číslo, okná ohodnotené hodnotou skóre MAX_SCORE a menej sa budú považovať za okná nekódujúcej oblasti

- dvojica INFORMANT_MATRIX a INFORMANT_WEIGHT
  Pre každého informanta INFORMANT, ktorého zarovnanie chceme zohľadniť pri predikcii, obsahuje hodnota INFORMANT_MATRIX cestu k skórovacej tabuľke. Hodnota premennej INFORMANT_WEIGHT určuje váhu daného informanta pri predikcii.

```
# window size
WINDOW = 92

# X
MIN_SCORE = 1.4
# Y
MAX_SCORE = -0.3

DROERE2_WEIGHT = 1
DROERE2_MATRIX = matrix1.txt

DROSIM1_WEIGHT = 0.8
DROSIM1_MATRIX = matrix2.txt

DROANA3_WEIGHT = 0.7
DROANA3_MATRIX = matrix3.txt

DP4_WEIGHT = 1
DP4_MATRIX = matrix4.txt
```

Obr. 3.3: Ukážka konfiguračného súboru

3.2 Popis algoritmu

Po úspešnom načítaní konfiguračného súboru je potrebné načítať vstupný FASTA súbor, ktorý obsahuje niekoľko sekvencií (viď. 3.1.1). Každá sekvencia sa následne samostatne spracúva, preto sa budeme ďalej v texte venovať len spracovaniu jednej sekvencie. Nižšie uvedený postup sa následne aplikuje na každú sekvenciu jednotlivo.

Označme si spracovávanú sekvenciu písmenom S a nech má dĺžku n. Teda

$$S = \{s_i\}_{i=1}^{n}$$

a je to určitá časť DNA referenčného organizmu, ktorého názov je zakódovaný v mene sekvencie, v TBA formáte (3.1.1).

Samotné spracovanie S možno rozdeliť do dvoch častí. V prvej časti programu sa pre sekvenciu S získa tabuľka zarovnaní informantov ku sekvencii S. V druhej časti sa na nej následne vyrátajú štatistiky podobné štatistikám z článku Lin a kol.[1], ktoré sme opísali v sekcii 2.4.3. Podľa nich sa potom niektoré časti S prehlásia za kódujúce alebo nekódujúce úseky.

3.2.1 Získanie zarovnania

Cieľom prvej časti je získať ku sekvencii S zarovnania sekvencií od informantov uvedených v konfiguračnom súbore (3.1.5).

V mene sekvencie je zakódovaná jej poloha v genóme. Podľa nej vieme vyextrahovať

príslušnú časť zarovnania z MAF súboru. MAF súbor pozostáva z blokov, ktorých tvar sme presne popísali v sekcii 3.1.2. V súbore teda nájdeme prvý relevantný blok, v ktorom začína zarovnanie pre S a postupne z ďalších blokov získavame zarovnania až kým nenaplníme celú tabuľku zarovnaní pre sekvenciu S. Týmto spôsobom môžeme postupovať, lebo bloky v MAF súbore sú utriedené podľa začiatku (viď. popis MAF súboru v sekcii 3.1.2).

Náš program vytvorí tabuľku, ktorej každý riadok predstavuje zarovnanie pre konkrétny organizmus. Je dôležité si uvedomiť, že všetky riadky tabuľky sú rovnako dlhé, ale zároveň môžu byť dlhšie než samotná sekvencia S, lebo do sekvencie S môže byť potrebné povkladať pomlčky na miestach, kde nastala delécia v sekvencii referenčného organizmu alebo inzercia v sekvencii niektorého informanta.

Pri vypĺňaní tabuľky so zarovnaniami sme museli riešiť niekoľko problémov, z ktorých niektoré uvedieme:

- Súbor s viacnásobnými celogenómovými zarovnaniami je rozdelený do blokov, ktoré ale nemusia nutne pokrývať celú sekvenciu S. Teda pre sekvenciu S môžu existovať úseky, pre ktoré zarovnanie v MAF súbore neexistuje. Tieto úseky sme v našej tabuľke vyplnili znakom „-“, ktorý v zarovnaniach štandardne predstavuje symbol pre inzerciu alebo deléciu.

- V jednom bloku sa nemusia nachádzať zarovnania všetkých informantov. Teda samotné zarovnanie informanta I nemusí pokrývať všetky úseky sekvencie S, ktoré sú pokryté MAF súborom. Táto situácia bola taktiež vyriešená znakom „-“ pre nedefinované úseky zarovnania informanta I.

Výsledkom je vyplnená tabuľka so zarovnaniami informantov ku sekvencii S, podľa ktorej sa môže v ďalších krokoch realizovať samotný výpočet.

3.2.2 Výpočet skóre kodónu

Po vytvorení tabuľky so zarovnaniami pre sekvenciu S sa spustí druhá časť programu, ktorej cieľom je rozhodnúť, či možno niektoré časti S považovať za kódujúce alebo nekódujúce. Základným krokom nášho algoritmu je priradiť každej trojici písmen v S skóre, ktoré vyjadruje, či sa môže jednať o kódujúcu alebo nekódujúcu oblasť. V nasledujúcom texte opíšeme, ako toto priradenie realizujeme.

Nech A je nejaký potenciálny kodón zo sekvencie S, tj. A je trojprvková súvislá podpostupnosť báz postupnosti S. Z tabuľky zarovnaní, ktorej vytvorenie sme opísali v predchádzajúcej sekcii, vieme pre konkrétneho informanta I vybrať trojprvkovú podpostupnosť hodnôt $B_I \in \{A, C, G, T, N, -\}$, ktorá prislúcha potenciálnemu kodónu A. Teda v tabuľke so zarovnaniami vyberieme z riadku informanta I hodnoty, ktoré sú v tom istom stĺpci ako jednotlivé bázy kodónu A.

Základnou časťou myšlienky nášho algoritmu je práve priradenie skóre $Sub(A, B_I)$ substitúcii medzi A a B_I. Pri priraďovaní tohto skóre postupujeme ako v článku Lin a kol.[1]. Rozlišujeme štyri možnosti:

- u informanta I došlo k inzercii
  Môže nastať prípad, že u informanta I došlo na mieste práve spracovávaného kodónu A k inzercii (obr. 3.4). To znamená, že na danom mieste nastala veľká zmena a preto tento prípad považujeme v kódujúcej oblasti za nežiaduci. Preto pre informanta I priradíme hodnotu $Sub(A, B_I) := 0$

- B_I obsahuje znak „-“
  Na pozícii kodónu A mohlo u informanta I dôjsť k delécii. V tomto prípade priradíme $Sub(A, B_I) := 0$

- aspoň jeden z A alebo B_I obsahuje znak „N“
  Pri sekvenovaní referenčného organizmu alebo informanta mohlo dôjsť k chybe. Nevieme teda aká báza sa na danej pozícii nachádza, priradíme $Sub(A, B_I) := 0$

- A aj B_I obsahujú len znaky báz „A“, „G“, „C“, „T“
  V tomto prípade sú A aj B_I regulérne kodóny a hodnotu pre túto substitúciu vieme zistiť zo skórovacej tabuľky (3.1.4). Skóre $Sub(A, B_I)$ získame vynásobením tohto čísla príslušnou váhou pre informanta I, ktorú sme získali z konfiguračného súboru (3.1.5).

Referenčný organizmus	T - - G C
Informant	T A G G A

(a) Došlo k inzercii u informanta
Priradí sa skóre nula

Referenčný organizmus	T - - G C
Informant	T - - G A

(b) Nedošlo k inzercii
Skóre sa priradí štandardne

Obr. 3.4: Inzercia v kodóne

Výsledné celkové skóre $Score(A)$ kodónu A následne vypočítame ako priemerné skóre všetkých informantov. Teda

$$Score(A) = \frac{\sum_{k=1}^{m} Sub(A, B_{I_k})}{m},$$

kde m je počet informantov.

Podobným spôsobom skórujeme potenciálny kodón i na komplementárnom vlákne. Tento potenciálny kodón i k nemu prislúchajúce B_I vytvoríme jeho komplementom a otočením (viď. 1.1).

3.2.3 Výpočet skóre okna a predikcia

V predchádzajúcej časti sme opísali, ako priradiť potenciálnemu kodónu skóre. Jedna trojica báz ale nemá dostatočne veľkú výpovednú hodnotu o tom, či sa nachádza v kódujúcej alebo nekódujúcej oblasti. Chceme ich preto spájať do väčších úsekov, aby sme vedeli rozhodnúť jednoznačne. Veľkosť týchto úsekov je nastavená v konfiguračnom súbore premennou WINDOW (viď. 3.1.5).

Nech S_W^i je súvislá podpostupnosť postupnosti S, pričom S_W^i má dĺžku W a začína na i-tej pozícii postupnosti S. Našim cieľom je prideliť každej súvislej podpostupnosti S_W^i skóre pre všetkých 6 čítacích rámcov. Toto skóre bude priemerom hodnôt skóre jednotlivých kodónov tejto postupnosti pre daný čítací rámec. Hodnota parametra W bola nastavená podľa hodnoty premennej WINDOW získanej z konfiguračného súboru. Pre zjednodušenie budeme v nasledujúcom texte podpostupnosť S_W^i označovať pojmom *okno* a W bude predstavovať *veľkosť okna*.

Okná v našom programe skórujeme iteratívne zľava doprava, pričom kvôli efektívnosti pri spracovaní okna začínajúceho na pozícii i použijeme časť hodnôt vypočítaných pre okno pred ním. Najprv si povieme, čo všetko je potrebné pre vypočítanie skóre jedného okna S_W^i a potom vysvetlíme toto zrýchlenie.

Spracovanie jedného okna

K tomu, aby sme vedeli oknu prideliť skóre, potrebujeme pre dané okno vyextrahovať príslušnú časť tabuľky so zarovnaniami. Ako sme už vysvetlili v sekcii 3.2.1, táto časť tabuľky zarovnaní môže mať dĺžku väčšiu než W, lebo u niektorého z informantov mohlo na mieste okna dôjsť k inzercii. Pre jednotlivé bázy okna S_W^i teda potrebujeme určiť ich pozíciu (číslo stĺpca) v tabuľke so zarovnaniami k S. Tieto pozície si uložíme do poľa pozícií p, pričom $p[j]$ je pozícia j-tej bázy v tabuľke zarovnaní. Kodón referenčného organizmu začínajúci na pozícii j v sekvencii S je teda v tabuľke postupnosť báz na pozíciách $p[j]$, $p[j+1]$ a $p[j+2]$.

Druhý typ informácie potrebnej k ohodnoteniu okna je pridelenie skóre kodónom okna S_W^i v jednotlivých čítacích rámcoch. Vytvorili sme si teda polia *cod_score* a *revcod_score*, v ktorých budeme uchovávať hodnoty skóre pre kodóny čítacích rámcov, pričom $cod\_score[j]$ je skóre kodónu začínajúceho v S na pozícii j a $revcod\_score[j]$ skóre kodónu reverzného komplementárneho ku kodónu na pozícii j.

Ako bolo povedané v časti 3.1.5, dĺžka okna je $W = 3k + 2$, pričom k je počet kodónov v okne S_W^i vo všetkých čítacích rámcoch. Kodóny okna S_W^i prislúchajúce čítaciemu rámcu $r = 0$ začínajú v sekvencii S na pozíciách $i, i+3, i+6, ...i+3(k-1)$. Kodóny prislúchajúce čítaciemu rámcu $r = 1$ kodóny na pozíciách $i+1, i+4, i+7, ...i+r+3(k-1)$ a čítaciemu rámcu $r = 2$ kodóny začínajúce v S na pozíciách $i+2, i+5, i+8, ...i+r+3(k-1)$. Pre reverzné čítacie rámce 3,4,5 vytvoríme kodóny

z kodónov čítacích rámcov 0,1,2 [2] ako reverzné komplementárne (viď 1.1).

Finálne skóre $score^r_{S^i_W}$ okna S^i_W pre čítací rámec r vypočítame ako priemerné skóre kodónov v danom čítacom rámci r.
Teda pre čítacie rámce $r \in \{0, 1, 2\}$ platí

$$score^r_{S^i_W} = \frac{\sum_{j=0}^{k-1} cod\_score[i + r + 3k]}{k}$$

A pre reverzné čítacie rámce $r \in \{3, 4, 5\}$ platí

$$score^r_{S^i_W} = \frac{\sum_{j=0}^{k-1} revcod\_score[i + (r - 3) + 3k]}{k}$$

Efektívne posúvanie okna

Vyššie uvedeným postupom vieme vypočítať prvky poľa pozícií p, prvky polí cod_score a $revcod\_score$ a taktiež výsledné skóre $score^r_{S^0_W}$ prvého okna S^0_W pre všetky čítacie rámce r. Pre ďalšie okná túto informáciu znovu použijeme.

Ak máme vyrátané skóre pre okno S^i_W, tak ďalšie okno, ktoré potrebujeme spracovať je okno S^{i+3}_W. Pre každý čítací rámec platí, že kodóny okna S^{i+3}_W vznikli odstránením prvého kodónu v S^i_W a pridaním jedného kodónu na koniec. Teda polia p, cod_score a $revcod\_score$ nemusíme celé nanovo vypĺňať, ale stačí len upraviť ich hodnoty zo spracovania predchádzajúceho okna. V poliach odstránime prvé tri hodnoty a dopočítame posledné tri. V našom programe sme teda na všetky tri polia 3-krát aplikovali shift poľa doľava a na koniec pridali novovypočítané hodnoty. Časová zložitosť tejto operácie v programovacom jazyku Perl, v ktorom je náš program naprogramovaný, nezávisí od dĺžky poľa, ale sa vykoná v konštantnom čase, a preto sme si jej použitie mohli dovoliť.

Taktiež celkové skóre okna $score^r_{S^{i+3}_W}$ vieme vypočítať z hodnoty skóre predchádzajúceho okna $score^r_{S^i_W}$. Preto sa teda celé spracovanie okna realizuje v konštantnom čase.

Predikcia

Každému čítaciemu rámcu r okna S^i_W sme teda priradili skóre $score^r_{S^i_W}$, ktoré predstavuje priemerné skóre kodónu v danom okne a čítacom rámci. Následne prebieha rozhodovanie, či možno okno S^i_W považovať za oblasť kódujúcu, nekódujúcu, alebo za oblasť, pre ktorú nevieme rozhodnúť. Citlivosť predikcie je regulovaná parametrami MIN_SCORE a MAX_SCORE z konfiguračného súboru.

[2] V tomto poradí, tj. z kodónov pre čítací rámec 0 kodóny rámca 3, z kodónov pre čítací rámec 1 kodóny rámca 4 atď.

Oblasť S_W^i považujeme v čítacom rámci r za kódujúcu, ak je skóre tejto oblasti aspoň také ako MIN_SCORE. Na druhej strane, ak je skóre vo všetkých čítacích rámcoch menšie alebo rovné MAX_SCORE, tak sa oblasť považuje za nekódujúcu. Následne vypíšeme na výstup predikciu v GTF formáte (3.1.3), pričom jednotlivé položky nastavíme v požadovanom tvare.

$$< start > := i$$
$$< end > := i + W - 1$$

- $score^r_{S_W^i} \geq$ MIN_SCORE – kódujúca oblasť
  $< feature >:= CDS$
  $< score >:= score^r_{S_W^i}$
  Položky $< strand >$ a $< frame >$ sa nastavia na požadované hodnoty podľa r.

- $\forall r : score^r_{S_W^i} \leq$ MAX_SCORE – nekódujúcu oblasť
  $< feature >:= noncoding$
  $< score >:= max\{score^r_{S_W^i} \mid r \in \{0, 1, 2, 3, 4, 5\}\}$
  Položky $< strand >$ a $< frame >$ zostanú nedefinované (symbol „.“).

Po vykonaní horeuvedeného postupu pre všetky okná sekvencie S je výstupom zoznam všetkých nájdených kódujúcich a nekódujúcich oblastí S v GTF formáte.

3.3 Rozšírenia základného algoritmu

Okrem samotného algoritmu na predikciu kódujúcich a nekódujúcich oblastí sme do nášho programu implementovali aj metódy, ktoré zabezpečujú zrýchlenie programu, alebo upravujú výstup do lepšej podoby.

3.3.1 Indexovanie

Jeden MAF súbor s viacnásobnými celogenómovými zarovnaniami je pomerne veľký. Ako príklad uvedieme, že pre nami testovaný organizmus *Drosophila melanogaster* má MAF súbor pre jeden chromozóm veľkosť väčšiu než 500MB.

Ak vstupný FASTA súbor obsahuje m sekvencií, tak pre každú z nich musíme v MAF súbore nájsť zarovnanie. Napriek informácii, že bloky v MAF súbore sú usporiadané podľa začiatku, to v najhoršom prípade znamená, že budeme musieť prečítať celý súbor až m-krát.

Z tohto dôvodu sme sa rozhodli pre každý MAF súbor vytvoriť indexový súbor, ktorý pozostáva z dvoch stĺpcov. V prvom stĺpci sa nachádza pozícia x v sekvencii referenčného genómu a v druhom stĺpci pozícia p v MAF súbore, na ktorú môžeme skočiť, ak chceme načítať zarovnanie pre sekvenciu, ktorá začína najskôr na pozícii x.

Hodnota p ukazuje na prvý blok v MAF súbore, ktorý končí za hranicou x. Pozície x vypisujeme s krokom $k = 5000$, tj. súbor obsahuje riadky pre $x = 0, k, 2k, ...$

Ak sme teda z hlavičky vo FASTA súbore zistili, že sekvencia S začína na pozícii x', tak nám stačí začať čítať MAF súbor od pozície p, ktorá je uvedená v indexovom súbore pri čísle $x = x' - (x' \bmod k)$, tj.

$$x = ki_{max};\ i_{max} = \max\{i \mid ki \leq x'\}$$

Indexový súbor vytvoríme jedným prechodom MAF súboru pri jeho prvom použití, pri ďalších použitiach binárnym vyhľadávaním nájdeme vhodnú dvojicu (x, p) a čítame z MAF súboru od pozície p až kým nenájdeme prvý blok obsahujúci začiatok S. Týmto mechanizmom namiesto čítania celého MAF súboru potrebujeme prečítať iba tie bloky, ktoré sa buď prekrývajú so skúmanou sekvenciou S alebo s oknom dĺžky najviac k pred jej začiatkom.

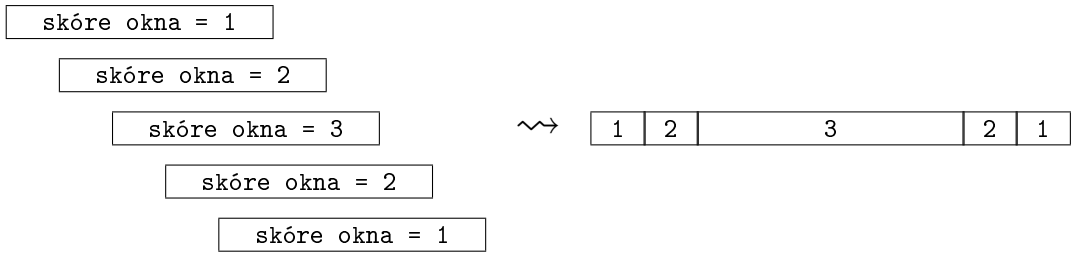
3.3.2 Spájanie intervalov

V sekcii 3.2.3 sme opísali, ako sa pre okno rozhodne, či ho možno považovať za kódujúce alebo nekódujúce. Následne sme informácie o okne spolu s hodnotou skóre vypísali na výstup v GTF formáte. Nevýhodou tohto riešenia je, že sa vypisuje vždy nový záznam o každom okne S_W^i. Rozhodli sme sa teda implementovať algoritmus, ktorý bude prekrývajúce sa okná v rovnakom čítacom rámci spájať a vypíše v GTF formáte informácie pre výsledné dlhšie okno, ktoré vzniklo zjednotením týchto okien. Najhoršie skóre spomedzi všetkých prekrývajúcich sa okien, bude predstavovať aj finálne skóre pre výsledné okno (aby sa dodržalo pravidlo, že všetky okná majú aspoň také skóre).

Rozdelili sme si teda jednotlivé okná (identifikované začiatkom, koncom, čítacím rámcom a hodnotou skóre) do šiestich skupín podľa čítacích rámcov, pričom v každej skupine boli usporiadané podľa začiatku (v tomto poradí boli získavané, takže ich nebolo nutné triediť). Na každú skupinu sme následne aplikovali rovnaký postup, pri ktorom sme postupne spracúvali jednotlivé okná v poradí. Po načítaní prvého okna O sme vytvorili pomocné pole p, ktoré malo dĺžku rovnú dĺžke okna O a do jednotlivých políčok $p[i]$ sme uložili hodnotu skóre okna O. Potom sme načítali ďalšie okno O' a overili, či sa prekrýva s oknom O.

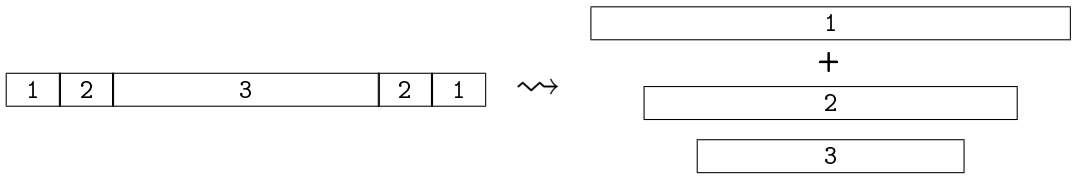
- Ak áno, tak sme zväčšili dĺžku poľa p o vyčnievajúci úsek a upravili prislúchajúce hodnoty $p[i]$ pre okno O' na maximum pôvodnej hodnoty a hodnoty skóre O'. Následne sme okno zjednotenia O a O' označili ako O a pokračovali načítaním nasledujúceho okna O' (obr. 3.5).

- Ak nie, tak sme jednotlivé údaje o okne zjednotenia O a hodnoty $p[i]$ uložili a postupovali, akoby sme práve načítali prvé okno.



Obr. 3.5: Spájanie prekrývajúcich sa okien v rovnakých čítacích rámcoch do okna zjednotenia. Výsledkom je pole s maximálnymi hodnotami skóre pre jednotlivé úseky a informácia o pozícii okna zjednotenia.

Takto sme z viacerých prekrývajúcich sa okien vytvorili jedno okno zjednotenia, ktorému sme pridelili najhoršie skóre, ktoré obsahuje a vypísali na výstup v GTF formáte. Ak by sa ale v tomto okne nachádzala aj oblasť s veľmi vysokým skóre, tak by sme túto informáciu stratili. Preto sme ešte následne postupne prechádzali poľom p od každej pozície j a vypísali úsek, v ktorom hodnota neklesla pod $p[j]$ (obr. 3.6). Čítanie od pozície j sme preskočili, ak $p[j] \leq p[j-1]$, lebo prípadné nájdené okno už bolo zahrnuté v predchádzajúcom. Týmto postupom sme efektívne vyriešili problém mnohých prekrývajúcich sa okien a nestratili sme informáciu o lepších oknách.



Obr. 3.6: Príklad zoznamu intervalov vypísaných do GTF súboru pre jedno okno zjednotenia. Údaj o samotnom okne zjednotenia a jeho rozdelenie na jednotlivé úseky za účelom poskytnutia informácie aj o častiach s vyšším skóre.

Kapitola 4

Výsledky

Po doplnení nášho modulu do programu ExonHunter, ktorý slúži ako jeho radca, sme sa ho rozhodli testovať na testovacích sekvenciách. Aktuálna anotácia genómu octovej mušky *Drosophila melanogaster* sa v súčastnosti považuje za pomerne presnú a preto sme mohli úspešnosť našich výsledkov vyhodnotiť. Samotnému testovaniu presnosti predikcií predchádzal výber vhodných konfiguračných nastavení, ktorý môže výrazne ovplyvniť namerané výsledky.

4.1 Použité dáta

V našej práci sme používali dve rozdielne sady dát, trénovacie a testovacie, ktoré sme získali z databázy UCSC Genome Browser [10]. Obe sady obsahovali niekoľko DNA sekvencií dĺžky dva milióny báz, ktoré pochádzali od octovej mušky *Drosophila melanogaster* z chromozómov chr2L a chr2R. Trénovacie dáta pozostávali spolu z pätnástich sekvencií (šesť z chromozómu chr2L a deväť z chromozómu chr2R). Testovacie dáta boli tvorené ôsmimi sekvenciami (šesť z chromozómu chr2L a dve z chromozómu chr2R). Referenčné anotácie pre trénovacie i testovacie sekvencie sme získali z databázy *RefSeq* umiestnenej na NCBI [13].

Viacnásobné celogenómové zarovnania sme taktiež získali z UCSC Genome Browser a boli vytvorené programom Multiz [9]. K referenčnému genómu mušky *Drosophila melanogaster* boli zarovnané genómy nasledujúcich druhov hmyzu: *Drosophila simulans, Drosophila sechellia, Drosophila yakuba, Drosophila erecta, Drosophila ananassae, Drosophila preudoobscura, Drosophila persimilis, Drosophila virilis, Drosophila mojavensis, Drosophila grimshawi, Drosophila willistoni, Anopheles gambiae, Anopheles mellifera, Tribolium castaneum.*

4.2 Voľba konfiguračných nastavení

V konfiguračnom súbore si užívateľ nastavuje parametre, ktoré budú ovplyvňovať výpočet nášho programu. Užívateľ musí taktiež vytvoriť skórovaciu tabuľku, podľa ktorej sa priradí substitúcii kodónov hodnota skóre. V nasledujúcom texte sa budeme venovať tomu, aké nastavenia sme používali pri testovaní a ako sme ich volili.

4.2.1 Skórovacia tabuľka

Keďže Lin a kol.[1] nesprístupnili skórovaciu tabuľku, s ktorou svoje predikcie vykonávali, museli sme si vytvoriť vlastnú. Pre naše testovacie účely sme si vytvorili len veľmi jednoduchú tabuľku, ktorá bola pre všetkých informantov jednotná. Taktiež sme ignorovali príbuznosti jednotlivých organizmov a preto sme jej váhu pre všetky organizmy zvolili 1.

Samotnú tabuľku hodnôt skóre pre substitúcie kodónov sme vytvorili podľa jednoduchých závislostí. Nech je kodón A substituovaný kodónom B a $Sub(A, B)$ je hodnota skóre priradená tejto substitúcii. Rozlišovali sme štyri možnosti:

- ak $A = B$, tak $Sub(A, B) := 0$
  Ak sú dva zarovnávané kodóny zhodné, nedošlo k žiadnej substitúcii a preto nemáme informáciu o tom, či ide o kódujúci alebo nekódujúci úsek. Z toho dôvodu sme v tomto prípade zvolili skóre pre substitúciu rovné nule.

- ak A kóduje rovnakú aminokyselinu ako B, tak $Sub(A, B) := 10$
  Ak substitúcia kodónov nemení kódovaná aminokyselina, nezmení sa ani výsledná bielkovina, ktorá bola pomocou nej kódovaná. Dá sa preto predpokladať, že sa jedná o kódujúcu oblasť. Preto sme zvolili kladné skóre.

- ak A kóduje inú aminokyselinu ako B, tak $Sub(A, B) := -10$
  Zmenou aminokyseliny v bielkovine sa väčšinou mení i tvar bielkoviny a teda i jej funkcia. Preto sa takéto substitúcie dejú v kódujúcich oblastiach menej a sú bežné pre nekódujúce oblasti. Zvolili sme preto pre túto možnosť skóre záporné.

- ak práve jeden z A a B je Stop kodón, tak $Sub(A, B) := -50$
  Zmenou jedného kodónu kódujúceho aminokyselinu na Stop kodón (a naopak) sa mení i dĺžka kódovanej aminokyseliny a preto sú takéto substitúcie v kódujúcich oblastiach zväčša veľmi nežiadúce. Voľba ešte menšieho záporného čísla sa intuitívne ukazuje správna.

4.2.2 Veľkosť okna a prahy citlivosti

S voľbou veľkosti okna a hodnotami MAX_SCORE a MIN_SCORE je to menej intuitívne a ich hodnoty môžu v istej miere závisieť od konkrétneho referenčného organizmu.

Napríklad ideálne zvolená veľkosť okna priamo závisí od dĺžky kódujúcich oblastí pre daný organizmus. Rozhodli sme sa teda testovať rôzne nastavenia programu na trénovacích sekvenciách.

Pri zisťovaní vhodných nastavení hodnôt parametrov WINDOW, MAX_SCORE a MIN_SCORE sme sa zaujímali o senzitivitu a špecificitu našich predikcií pre jednotlivé bázy skúmanej sekvencie, pričom sme používali štandardné definície:

$$\text{senzitivita} = \frac{\#\text{ správne predikovaných}}{\#\text{ správne predikovaných} + \#\text{ nepredikovaných}}$$

$$\text{špecificita} = \frac{\#\text{ správne predikovaných}}{\#\text{ správne predikovaných} + \#\text{ nesprávne predikovaných}}$$

V prípade skúmania kódujúcich úsekov sme bázu považovali sa *správne predikovanú*, ak ju náš program prehlásil v konkrétnom čítacom rámci r a na konkrétnom vlákne v (+ alebo -) za bázu v kódujúcej oblasti a zároveň táto báza aj skutočne bola v rámci r a na vlákne v kódujúca. Za *nesprávne predikovanú* bola považovaná báza, ktorú program prehlásil v čítacom rámci r a na vlákne v za kódujúcu, no v skutočnosti sa v kódujúcej oblasti nenachádzala, alebo nachádzala, ale v inom čítacom rámci alebo na inom vlákne. Kategóriu *nepredikovaných báz* tvorili všetky tie bázy, ktoré sa nachádzali v kódujúcich oblastiach, no náš program ich nepredikoval, alebo predikoval nesprávne. V prípade pozorovania nekódujúcich úsekov sme postupovali analogicky.

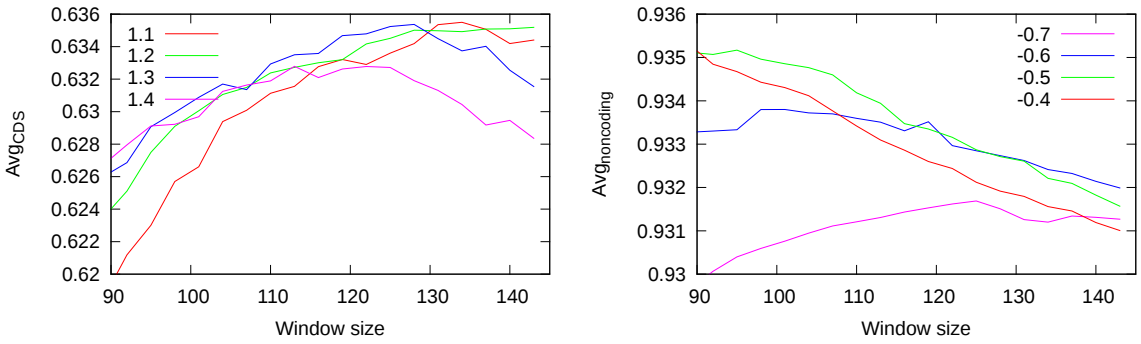
Z významu MIN_SCORE a MAX_SCORE a definovania senzitivity a špecificity je zrejmé, že sme sa mohli samostatne zaoberať predikciou kódujúcich a nekódujúcich oblastí. Program sme spúšťali pre rôzne nastavenia parametrov WINDOW, MIN_SCORE a MAX_SCORE a vyrátali sme hodnoty:
$Sn_{CDS}(\text{WINDOW}, \text{MIN\_SCORE})$, $Sp_{CDS}(\text{WINDOW}, \text{MIN\_SCORE})$
$Sn_{noncoding}(\text{WINDOW}, \text{MAX\_SCORE})$, $Sp_{noncoding}(\text{WINDOW}, \text{MAX\_SCORE})$

Pri hľadaní vhodných nastavení sme sledovali priemerné hodnoty senzitivity a špecificity. Teda hodnoty

$$Avg_{CDS} := \frac{Sn_{CDS} + Sp_{CDS}}{2}$$

$$Avg_{noncoding} := \frac{Sn_{noncoding} + Sp_{noncoding}}{2}$$

Na obrázku 4.1a sú znázornené priemerné hodnoty kódujúcich oblastí pre niektoré nastavenia MIN_SCORE. Podobne na obrázku 4.1b sa nachádzajú priemerné hodnoty nekódujúcich oblastí pre nastavenia MAX_SCORE. Pomocou týchto pozorovaní sme zvolili nastavenia parametrov WINDOW = 128, MIN_SCORE = 1.2 a MAX_SCORE = -0.5, ktoré by mali dosahovať dobré výsledky i na testovacích DNA sekvenciách mušky *Drosophila melanogaster*.



(a) Kódujúce oblasti

(b) Nekódujúce oblasti

Obr. 4.1: Priemerné hodnoty senzitivity a špecificity pre rôzne hodnoty MIN_SCORE a MAX_SCORE uvedené v legende tabuliek.

4.2.3 Využitie informácie ExonHunter-om

Následne prebehlo trénovanie programu ExonHunter, ktoré má za úlohu zistiť pravdepodobnosť, že je radca správny a podľa nej ho budú rady nášho radcu ovplyvňovať. Na trénovanie bol použitý výsledný GTF súbor s výsledkom nášho algoritmu pre trénovacie sekvencie s nastaveniami, ktorých získanie sme opísali v predchádzajúcom texte. Intervaly z nášho GTF súboru boli spracovávané existujúcimi skriptami v programe ExonHunter a po natrénovaní bol program ExonHunter schopný využiť nášho radcu pri svojich predikciách.

4.3 Analýza predikcií

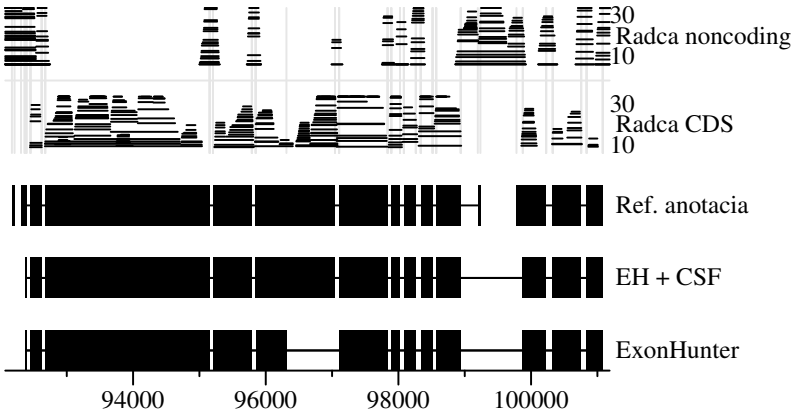
Po zvolení skórovacej tabuľky a konfiguračných parametrov, sme s týmito nastaveniami na testovacích sekvenciách testovali úspešnosť hľadania génov s využitím externej informácie z viacnásobných celogenómových zarovnaní a výsledky sme porovnávali s referenčnou anotáciou. V tabuľke 4.1 sme porovnali program ExonHunter bez a následne s využitím externej informácie z viacnásobných celogenómových zarovnaní.

	EH	EH+CSF – pôvodné	EH+CSF – modifikované
Senzitivita génov	41.98%	31.24%	40.42%
Špecificita génov	47.63%	42.28%	46.15%
Senzitivita exónov	72.03%	61.32%	70.24%
Špecificita exónov	72.48%	72.81%	71.91%
Senzitivita nukleotidov	94.67%	84.26%	93.65%
Špecificita nukleotidov	91.49%	96.10%	92.63%

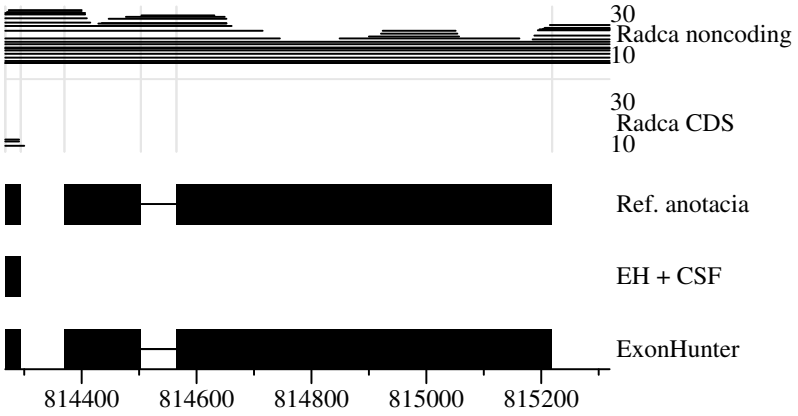
Tabuľka 4.1: Porovnanie predikcií programu ExonHunter bez využitia externej informácie a s využitím externej informácie z viacnásobných celogenómových zarovnaní (CSF). Porovnávané sú výsledky s pôvodnými nastaveniami a modifikovanými. V referenčnej anotácii bol celkový počet génov 1796, počet exónov 7681 a kódujúcich nukleotidov 3284776.

Na vyhodnocovanie predikcií sme použili program Eval [11] a skúmali sme senzitivitu a špecificitu predikcie génov, exónov a nukleotidov (jedna báza kódujúceho exónu). Prvok (gén, exón, nukleotid) sa považuje za *správne predikovaný*, ak sa jeho pozícia a typ presne zhoduje s prvkom v referenčnej anotácii. Za *nesprávne predikovaný* sa považuje prvok, ktorého typ a pozícia sa nezhodujú so žiadnym prvkom referenčnej anotácie a *nepredikované* prvky sú tie prvky referenčnej anotácie, pre ktoré v testovanej predikcii neexistuje žiaden prvok rovnakého typu a pozície.

Z porovnaní vidno, že predikcie s využitím nového radcu sa od pôvodných zhoršili vo všetkých mierach okrem špecificity exónov a kódujúcich nukleotidov. Neúspech predikcie sme analyzovali a vizualizovali pomocou nástroja Mikroskop [12] a zistili, že i keď sa nám pomocou externej informácie podarilo správne predikovať niektoré oblasti, ktoré bez jej využitia správne predikované neboli (obr. 4.2a), tak častokrát rady presnosť predikcií zhoršili. Vo veľa prípadoch to bolo zapríčinené tým, že radca označil aj kódujúce úseky za úseky nekódujúce a tým zapríčinil výslednú nesprávnu predikciu (obr. 4.2b). Rozhodli sme sa preto predikciu zopakovať so sprísnenou hranicou pre nekódujúce oblasti, zvolili sme MAX_SCORE $= -2$. Z pozorovaní sme taktiež zistili, že predikcie okrajových častí okien bývajú vo veľa prípadoch nepresné a preto sme modifikovali program ExonHunter tak, aby okrajové časti nami predikovaných intervalov orezával. Intervaly *CDS* sme orezali o 30 báz a intervaly *noncoding* o 200 báz z každého konca.



(a) Nové predikované exóny



(b) Zle predikované nekódujúce oblasti

Obr. 4.2: Vizualizácia predikcií niektorých úsekov testovacích sekvencií. Porovnávajú sa predikcie programu ExonHunter bez a s použitím externej informácie z celogenómových zarovnaní (CSF) a referenčnej anotácie. Zvýraznené časti predstavujú predikované exóny v jednotlivých anotáciách.

Úpravami sme získali výsledky, ktoré sa okrem špecificity exónov a kódujúcich nukleotidov v porovnaní s predchádzajúcim meraním zlepšili, no napriek tomu sme nezaznamenali zlepšenie v porovnaní s predikciou programu ExonHunter bez využitia našich rád (tab. 4.1).

4.4 Návrh zlepšenia

Z našich neuspokojivých výsledkov predikcie vyplýva, že zvolená jednoduchá skórovacia tabuľka k zaznamenaniu zlepšenia úspešnosti predikcií programu ExonHunter v súčasnosti nepostačuje. Je preto potrebné vytvoriť efektívnejší spôsob skórovania substitúcií kodónov pre jednotlivé organizmy.

Zo známych anotácií a zarovnaní sekvencií si môžeme požadovanú skórovaciu ta-

buľku vytvoriť. Podľa porovnania výskytov jednotlivých substitúcií kodónov v kódujúcich a nekódujúcich oblastiach dokážeme zistiť, či je daná substitúcia častejšia v úsekoch, ktoré kódujú nejakú bielkovinu, alebo nie. Čím väčšie množstvo anotovaných sekvencií máme k dispozícii, tým vierohodnejšiu skórovaciu tabuľku sme schopný vytvoriť. Na vytvorenie tabuľky môžeme použiť spôsob podobný prístupu Lin a kol., ktorý sme opísali v časti 2.4.3.

Záver

Cieľom našej práce bolo implementovať modul na použitie celogenómových zarovnaní v systéme na hľadanie génov - ExonHunter. Vytvorili sme radcu, ktorý na základe porovnaní DNA sekvencií príbuzných organizmov označil niektoré časti skúmanej sekvencie za kódujúce a nekódujúce a túto informáciu poskytol programu ExonHunter, ktorý ju následne využil pri predikcii génov.

Program s doplnenou možnosťou využitia viacnásobných celogenómových zarovnaní sme testovali na testovacích DNA sekvenciách mušky *Drosophila melanogaster*. Keďže sme použili len jednoduchú skórovaciu tabuľku, tak sa nám presnosť predikcií zlepšiť nepodarilo. Samotné rady síce boli pomerne presné, ale v spojení s programom ExonHunter k zlepšeniu nedochádzalo. Preto predpokladáme, že pomocou zložitejších postupov tvorby skórovacej tabuľky dôjde k zlepšeniu presnosti a náš modul bude možné v budúcnosti efektívne využiť.

Literatúra

[1] Lin a kol. 2007. Revisiting the protein-coding gene catalog of Drosophila melanogaster using 12 fly genomes. In *Genome Research.* roč. 17,č. 12,s. 1823-1836

[2] Lin a kol. 2008. Performance and Scalability of Discriminative Metrics for Comparative Gene Identification in 12 Drosophila Genomes. In *PLoS Comput Biol.* roč. 4,č. 4,s. e1000067

[3] Zvelebil, Marketa a Baum, Jeremy O. 2007. *Understanding bioinformatics.* Garland Science. ISBN 0-8153-4024-9

[4] Brejová, Bronislava. 2005. *Evidence Combination in Hidden Markov Models for Gene Prediction.* dizertačná práca : University of Waterloo, 2005

[5] Brejová a kol. 2005. ExonHunter: a comprehensive approach to gene finding. In *Bioinformatics.* roč. 21,č. 1,s.57-65

[6] Kováč, Peter. 2010. *Implementácia externých zdrojov dát v hľadaní génov.* bakalárska práca. Bratislava : Univerzita Komenského, 2010

[7] Rabiner, Lawrence R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE.* roč. 77, s.257-286

[8] Forney, G. David Jr. 1973. The Viterbi algorithm. In *Proceedings of the IEEE.* roč. 61,č. 3,s. 268-278

[9] Blanchette, M. a kol. 2004. Aligning multiple genomic sequences with the threaded blockset aligner. In *Genome Research.* roč. 14,s. 708-715

[10] Kent W.J a kol. 2002. The human genome browser at UCSC. In *Genome Research.* roč. 12,č. 6,s. 996-1006. `http://genome.ucsc.edu`

[11] Keibler, E. a Brent, M. R. 2003. Eval: A software package for analysis of genome annotations. In *BMC Bioinformatics*, roč. 4,č. 1

[12] Brejová, B. a Vinař, T. 2008. *Mikroskop, flexible tool for creating diagrams of sequence annotations.* `http://www.bioinformatics.uwaterloo.ca/downloads/mikroskop`

[13] Pruitt, K.D., Tatusova, T., Maglott, D.R. 2007. NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. In *Nucleic Acids Research.* roč. 35,č. Database issue,s. 61-65

[14] *GenBank.* `http://www.ncbi.nlm.nih.gov/sites/entrez?db=nucleotide`