Metódy v bioinformatike

Broňa Brejová a Tomáš Vinař

Fakulta matematiky, fyziky a informatiky
Univerzita Komenského v Bratislave
2011

Metódy v bioinformatike
Autori: Broňa Brejová a Tomáš Vinař
Vydavateľ: Knižničné a edičné centrum, Fakulta matematiky, fyziky a informatiky, Univerzita Komenského, Mlynská dolina, 842 48 Bratislava
Vydanie: prvé
Počet strán: 92
Miesto a rok vydania: Bratislava, 2011

Podporené grantovou agentúrou KEGA v rámci projektu 058-016UK-4/2010 "Medziodborové štúdium bioinformatiky: Koncepcia štúdia a realizácia pilotných predmetov"

ISBN 978-80-89186-94-5

Obsah

Predslov

Tieto skriptá pokrývajú materiál z vybraných prednášok predmetu Metódy v bioinformatike, ktoré na Fakulte matematiky, fyziky a informatiky Univerzity Komenského v Bratislave v školských rokoch 2010/11 a 2011/12 prednášali Broňa Brejová a Tomáš Vinař. Cieľom predmetu je oboznámiť študentov magisterského štúdia z biologických aj informatických odborov s princípmi základných bioinformatických techník. Prednášky, ktoré sú spoločné pre biológov a informatikov, zavedú modely a výpočtové problémy používané v jednotlivých oblastiach a načrtnú myšlienky výpočtových metód, ktorými sa tieto problémy riešia. Dôraz je tiež kladený na pochopenie obmedzení, ktoré jednotlivé metódy majú. Prednášky sú doplnené cvičeniami, ktoré sú ponúkané v dvoch verziách. Cieľom cvičení pre informatikov je prebrať základné pojmy molekulárnej biológie potrebné pre pochopenie predmetu, ako aj hlbšie detaily informatických metód spomenutých na prednáškach a ďalší doplňujúci materiál. Cieľom cvičení pre biológov je doplniť chýbajúce znalosti z informatiky a matematiky potrebné pre pochopenie prednášok a precvičiť použitie preberaných metód na reálne dáta.

V budúcnosti plánujeme skriptá rozšíriť o cvičenia a ďalšie prednášky. Čitateľ však už dnes môže nájsť materiály k cvičeniam a prednáškam na stránke predmetu `http://compbio.fmph.uniba.sk/vyuka/mbi/`.

Kapitola 1

Sekvenovanie a zostavovanie genómov

DNA sekvencie organizmov sú jedným z najdôležitejších zdrojov informácií, ktorými sa bioinformatika zaoberá. V tejto kapitole sa zoznámime s problematikou *sekvenovania*, teda procesom získavania DNA sekvencií. Technológie sekvenovania sú o to zaujímavejšie, že sa v nich spájajú experimentálne metódy so zložitou výpočtovou analýzou (tzv. *zostavovanie* genómu) a obe tieto zložky sú nevyhnutné pre úspešné získanie DNA sekvencie živého organizmu.

Históriu sekvenovania v krátkosti sumarizuje tabuľka 1.1. Začína sa v roku 1976, keď bol osekvenovaný prvý genóm jednoduchého RNA vírusu. Doteraz pravdepodobne najväčším úspechom v tejto oblasti bolo v roku 2001 osekvenovanie ľudského genómu. Tento veľkolepý projekt naštartovaný v roku 1988 postupne prerástol do širokej medzinárodnej spolupráce verejných inštitúcií. Projekt inšpiroval vývoj sekvenovacích technológií, ako aj alternatívne prístupy k sekvenovaniu, čo vyústilo v roku 1998, keď súkromná spoločnosť Celera Genomics ohlásila, že svoju verziu ľudského genómu vytvorí do roku 2001. Nakoniec boli obe verzie publikované simultánne v roku 2001 v časopisoch Nature (verejný projekt) a Science (súkromný projekt).

Sekvencia ľudského genómu obsahuje približne 3.2 miliardy báz a z informatického hľadiska ju môžeme jednoducho reprezentovať niekoľkými reťazcami zloženými zo znakov `A`, `C`, `G` a `T`, pričom každý takýto reťazec reprezentuje jeden ľudský chromozóm. Na získanie tejto reprezentácie ľudského genómu boli vynaložené vyše 3 miliardy amerických dolárov. Inovácie technológií však dnes umožňujú osekvenovať podobne veľký genóm s rádovo nižšími nákladmi (desiatky tisíc dolárov). Aj preto sme v súčasnosti svedkami veľkého množstva sekvenovacích projektov a stále aktívneho vývoja relevantných algoritmov.

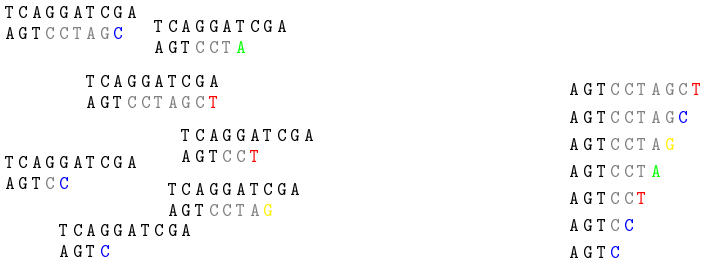
V tejto kapitole si ukážeme základné princípy sekvenovania a niekoľko algoritmov, ktoré sa používajú na následné zostavovanie genómov. V závere si ukážeme, ako nové sekvenovacie technológie rozširujú naše možnosti analýzy genómov, pričom z informatického hľadiska prinášajú nové zaujímavé problémy.

1.1 Sangerovo sekvenovanie

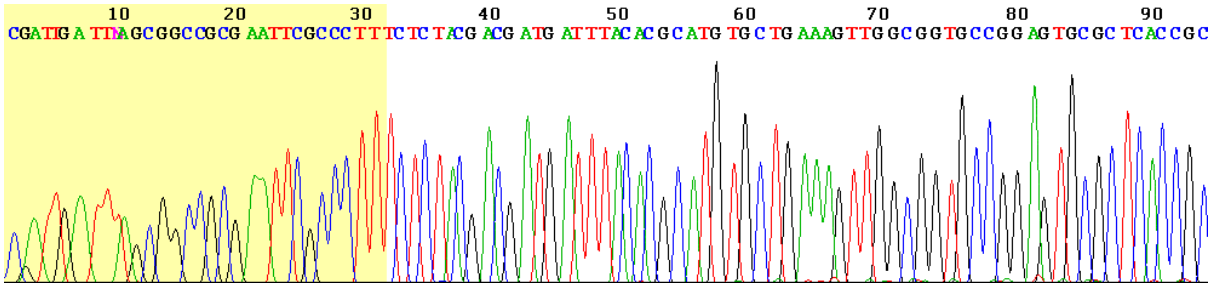
Sekvenovanie, tak isto ako veľa ďalších technológií molekulárnej biológie, je založené na kreatívnom využití enzýmov a biologických procesov, ktoré bežne prebiehajú v bunke. Na obrázku 1.1 je zobrazený spôsob, akým prebieha Sangerovo sekvenovanie – tradičná metóda použitá aj na sekvenovanie ľudského genómu.

Tabuľka 1.1: História sekvenovania (podľa Roberts (2001))

1976	MS2 (RNA vírus) 40 kB
1988	projekt sekvenovania ľudského genómu (15 rokov)
1995	baktéria H. influenzae 2 MB, shotgun (TIGR)
1996	S. cerevisiae 10 MB, BAC-by-BAC (Belgicko, Británia)
1998	C. elegans 100 MB, BAC-by-BAC (Wellcome Trust)
1998	Celera: ľudský genóm do troch rokov!
2000	D. melanogaster 180 MB, shotgun (Celera, Berkeley)
2001	2x ľudský genóm 3 GB (NIH, Celera)
po 2001	Myš, potkan, kura, šimpanz, pes, makak,...
2007	Watsonov a Venterov genóm (454)
čoskoro	1000 ľudských genómov, 22 cicavcov



Obr. 1.1: Schéma Sangerovho sekvenovania. Sekvenujeme DNA sekvenciu `AGCTAGGACT`, ktorú zobrazujeme sprava doľava. Použijeme primer `AGT` komplementárny ku koncu našej sekvencie. Pri sekvenovacej reakcii nám vzniknú segmenty rôznych dĺžok, každý ukončený modifikovaným a ofarbeným nukleotidom (vľavo). Na géli potom zoradíme segmenty podľa dĺžky a laserovým zariadením odčítame poradie farieb (modrá, modrá, červená, zelená, žltá, modrá, červená), ktoré premeníme na poradie nukleotidov na komplementárnom vlákne, teda `AGTCCTAGCT` (prvé `AGT` je primer).



Obr. 1.2: Sekvenovací profil (sequencing trace). Zdroj: wikipedia.org.

Dvojvláknová DNA sa najprv zahreje, čím sa rozdelí na jednovláknovú. V živej bunke je takúto jednovláknovú DNA možné použiť na vytvorenie nových kópií za predpokladu, že sú k dispozícii príslušné enzýmy a zásoba vhodných stavebných prvkov—nukleotidov. Pri sekvenovacej reakcii tento proces zmodifikujeme tak, že pridáme špeciálne nukleotidy, ktoré sú jednak zafarbené fosforeskujúcou farbou (každý nukleotid inou) a jednak sú zmodifikované takým spôsobom, aby kopírovanie DNA nemohlo po použití modifikovaných nukleotidov pokračovať.

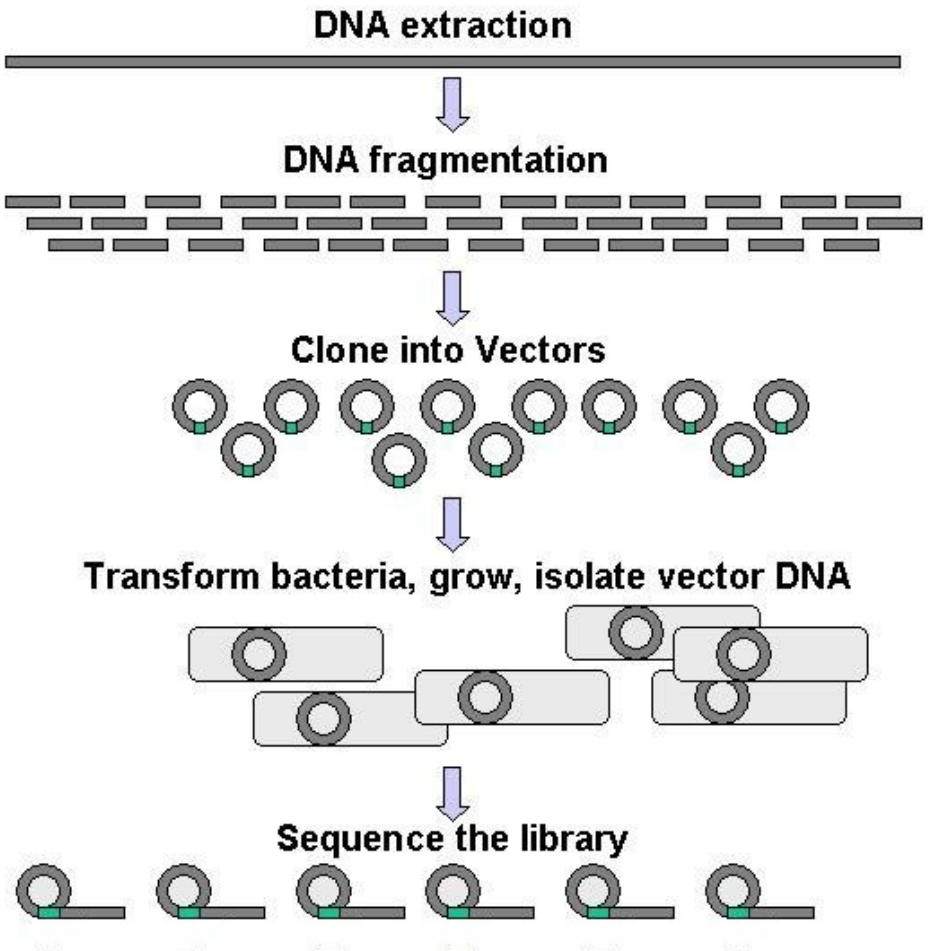
V takejto modifikovanej reakcii sa nám namiesto úplných komplementárnych kópií DNA vytvorí množstvo segmentov rôznych dĺžok, pričom každý segment bude vlastne predčasne ukončenou komplementárnou kópiou našej pôvodnej DNA. Každý takýto segment bude navyše ofarbený farbou, ktorá zodpovedá poslednému nukleotidu v danom segmente. Pri vhodnej koncentrácii modifikovaných nukleotidov budú výsledkom reakcie všetky takéto segmenty, t.j. každá pozícia v našej DNA sekvencii bude zastúpená nejakým segmentom.

Segmenty je teraz možné zoradiť podľa veľkosti pomocou elektroforézneho gelu a pomocou laserového zariadenia je možné prečítať postupnosť jednotlivých farieb, ktorá nám vlastne dáva poradie nukleotidov v sekvenovanej vzorke DNA.

Na obrázku 1.2 je zobrazený typický výsledok sekvenovania—sekvenovací profil (*sequencing trace*). Každý farebný vrchol zodpovedá jednému nukleotidu. Zo sekvenovacieho profilu však nie je možné vždy určiť každý nukleotid jednoznačne a správne. Na určenie jednotlivých nukleotidov a odhad skóre kvality sa u Sangerovho sekvenovania používa tradične program PHRED. Výstupom programu je pre každú pozíciu, kde je to možné určiť, báza `A`, `C`, `G`, alebo `T` a skóre kvality q, kde pravdepodobnosť, že báza bola určená chybne je ohraničená hodnotou $10^{-q/10}$; bázy s kvalitou $q > 40$ sú teda určené správne na 99.99%, kým z báz s kvalitou $q = 20$ je chybná približne každá stá báza. Podobný systém odhadu kvality báz je súčasťou prakticky všetkých sekvenovacích postupov.

Žiaľ, túto technológiu nie je možné aplikovať na ľubovoľne dlhé úseky DNA. V praxi Sangerovo sekvenovanie dokáže osekvenovať s dostatočnou spoľahlivosťou len cca. 500-1000 báz z konca vzorky DNA. Preto pri sekvenovaní genómov musíme použiť podstatne zložitejší postup (viď obrázok 1.3).

Viacero kopií DNA najskôr rozsekáme (napr. reštrikčným enzýmom alebo sonikáciou). Keďže DNA sa roztrhne na náhodných miestach, dostaneme množstvo menších navzájom prekrývajúcich sa fragmentov. Tieto fragmenty potom vložíme do umelo vytvoreného malého cirkulárneho chromozómu, ktorý nazývame vektor. Vektory je potom možné namnožiť vložením do baktérií a následne vyextrahovať. Takýmto spôsobom dostaneme tzv. *knižnicu* množstva malých segmentov DNA, ktorých konce potom môžeme osekvenovať



Obr. 1.3: Príprava knižnice na sekvenovanie. Zdroj: wikipedia.org.

postupom uvedeným vyššie.

V praxi teda dostaneme množstvo krátkych sekvencií dĺžky cca. 500-1000 báz, ktoré začínajú na náhodných miestach v genóme. Ako z takýchto malých kúskov dostaneme ucelený obraz o celom genóme? Týmto problémom sa budeme zaoberať v ďalšej kapitole.

1.2 Zostavovanie genómov (sequence assembly)

Žiadna zo súčasných sekvenovacích technológií nedokáže osekvenovať celý genóm, namiesto toho je výsledkom spústa menších fragmentov (*sequencing reads*), ktoré je potrebné zostaviť do súvislej sekvencie. Náročnosť zostavovania genómov závisí od niekoľkých faktorov:

- dĺžka jednotlivých fragmentov (čím dlhšie, tým lepšie),
- dĺžka zostavovanej sekvencie (čím kratšia, tým lepšie),
- priemerná hĺbka pokrytia – koľko fragmentov v priemere prekrýva konkrétnu pozíciu zostavovanej sekvencie (čím viac, tým lepšie).

Keďže zostavovanie genómov nie je vôbec jednoduchou úlohou, v počiatkoch sekvenovania sa na zvýšenie úspešnosti používala metóda *BAC-by-BAC*. Celý genóm sa najskôr rozdelí na prekrývajúce sa časti o veľkosti cca. 100–200 tisíc báz, ktoré sa tiež nazývajú BACy (bacterial artificial chromosome). Potom nasleduje náročný proces mapovania, kde sa experimentálnymi metódami určí približná poloha každého BACu v genóme a vyberie sa skupina BACov, ktoré pokrývajú celý genóm. Tieto sa potom jednotlivo sekvenujú. Výhodou takéhoto postupu je, že zostavujeme podstatne kratšie sekvencie. Nevýhodou je zdĺhavý a náročný proces mapovania.

V súčasnosti sa najčastejšie používa tzv. *shotgun* prístup, kde sa jednoducho rovno sekvenuje celý genóm s predpokladom, že bioinformatické metódy budú vedieť genóm poskladať aj za takýchto sťažených okolností. Vynechanie mapovania v tomto prípade výrazne znižuje cenu a zvyšuje rýchlosť sekvenovania, na druhej strane sa často stane, že genóm nie je možné poskladať až do tvaru celých chromozómov a sekvencia chromozómu tak zostane v tvare niekoľkých nespojených segmentov (tzv. *kontigov*).

V oboch prípadoch však z informatického hľadiska ide o tú istú úlohu: vstupom je množstvo krátkych prekrývajúcich sa segmentov sekvenovanej DNA, našim cieľom je zostaviť pôvodnú DNA sekvenciu. Ďalej si ukážeme niekoľko spresnení tohto problému, ako aj najčastejšie používané riešenia.

1.2.1 Najkratšie spoločné nadslovo

Asi najjednoduchšou formuláciou problému zostavovania genómov je teoretický problém hľadania *najkratšieho spoločného nadslova* (shortest common superstring).

Definícia 1 (Problém najkratšieho spoločného nadslova (shortest common superstring))**.** Je daných niekoľko reťazcov (osekvenovaných fragmentov) a úlohou je nájsť najkratší možný reťazec, ktorý obsahuje všetky fragmenty ako súvislé podreťazce.

Napríklad pre vstupné fragmenty `AAA`, `AAC`, `ACA`, `ACC`, `CAA`, `CAC`, `CCA` a `CCC` je najkratšie možné spoločné nadslovo `AAACCCACAA`.

Pojem najkratšieho nadslova je motivovaný nasledovne. V rámci definície problému nestačí povedať, že každý fragment musí byť podreťazcom výslednej sekvencie. To by sme totiž mohli úlohu riešiť triviálne: zreťazíme všetky fragmenty v ľubovoľnom poradí. To zjavne nie je zmysluplné riešenie nášho biologického problému. Intuitívne, cieľom je čo najviac využiť prekryvy medzi jednotlivými fragmentami: dva fragmenty, ktoré majú substanciálny prekryv sa pravdepodobne prekrývajú aj v sekvenovanej vzorke DNA. Požiadavka, aby nadslovo bolo najkratšie možné, je jednoduchý spôsob, ako takéto prekryvy zabezpečiť a súčasne nám toto kritérium dáva elegantný informatický problém, ktorý možno študovať aj z teoretického hľadiska.

Nanešťastie je tento problém NP-ťažký, takže nepoznáme žiadny rýchly algoritmus, ktorý vždy nájde najlepšie riešenie. Pre ilustráciu ukážeme kontrapríklad pre jednoduchú heuristiku, ktorá v každom kroku nájde dva fragmenty, ktoré sa prekrývajú najviac a zlúči ich do jedného segmentu.

Ak vezmeme fragmenty `CATATAT`, `TATATA` a `ATATATC`, tak touto heuristikou najprv spojíme fragmenty `CATATAT` a `ATATATC`, a potom k nim pripojíme fragment `TATATA`. Výsledné nadslovo má dĺžku 14 (`CATATATCTATATA`), optimálne riešenie však má dĺžku 10 (`CATATATATC`).

O našej jednoduchej heuristike je možné dokázať, že je to 3.5-aproximačný algoritmus: nájdené riešenie bude mať vždy dĺžku nanajvýš 3.5-násobku optimálnej dĺžky. Je otvoreným problémom, či je tento algoritmus aj 2-aproximačný algoritmus; existuje však iný 2.5-aproximačný algoritmus. V každom prípade ale aproximačné algoritmy nie sú pre tento problém príliš užitočné: aj keby sa nám podarilo nájsť 2-aproximačný algoritmus, nie je jasné, či riešenie, ktoré je dvakrát také dlhé ako optimálne, by v sebe obsahovalo informáciu užitočnú pre biológov.

1.2.2 Overlap–Layout–Consensus

Aj keď je problém najkratšieho spoločného nadslova pomerne elegantnou formuláciou problému zostavovania genómov, tento problém je príliš ťažký, aby bolo jeho riešenie praktické pre veľké genómy. Navyše existuje množstvo faktorov, ktoré táto formulácia neberie do úvahy:

- Pri sekvenovaní sa vyskytujú chyby. Sangerovo sekvenovanie momentálne urobí cca. 1 chybu na 100 báz. V praxi teda dostávame na vstupe fragmenty, ktoré sa od podslova pôvodnej sekvencie môžu mierne líšiť. Pri určovaní, či ide o skutočnú odlišnosť, alebo o sekvenovaciu chybu je navyše potrebné zohľadniť skóre kvality, ktoré sme spomínali vyššie.

- Aj v jedinom genóme môžu byť niektoré pozície polymorfné. Ľudský genóm obsahuje z každého chromozómu (okrem X a Y) dve kópie, jednu zdedenú od otca a druhú od matky. Tieto chromozómy sa môžu navzájom od seba líšiť: v ľudskom genóme nájdeme jednu polymorfnú pozíciu na približne 1000 báz. Pri sekvenovaní však vidíme fragmenty, ktoré sú zmesou z oboch chromozómov.

Obr. 1.4: **Spárované fragmenty (pair-end reads).** U Sangerovho sekvenovania s bakteriálnym klonovaním sa obvykle používajú segmenty dĺžky 2kbp (tzv. *plazmidy*) v kombinácií s dlhšími *kozmidmi* (cca. 40kbp). Pri sekvenovaní druhej generácie, kde sú výsledné prečítané fragmenty kratšie, sa často používa kombinácia knižníc so segmentami dĺžok od 200bp do 100kbp.

- Fragmenty na vstupe môžu byť kontaminované cudzou sekvenciou. Často napríklad vidíme sekvencie baktérií, ktoré sme použili na klonovanie. Typickou chybou sekvenovania sú tiež chimérické fragmenty, ktoré vznikli kombináciou dvoch nesúvisiach kúskov genómu.

- Typický genóm nie je jedna sekvencia, ale je organizovaný do niekoľkých chromozómov. Pokrytie genómu fragmentami môže byť navyše nerovnomerné, v niektorých častiach môže sekvencia úplne chýbať, čo môže znemožniť úplné poskladanie genómu.

- Častým úkazom v rámci genómov vyšších organizmov je tzv. repetitívna sekvencia. Sekvencie, ktoré sú jednoduchým mnohonásobným opakovaním niekoľkých nukleotidov tvoria až 50% ľudského genómu. Takéto sekvencie je veľmi náročné poskladať.

  Ak napríklad vidíme 10 fragmentov `TTAATA`, 10 fragmentov `ATATTA` a 3 fragmenty `TTAGCT`, tak najkratšie spoločné nadslovo je `TTAATATTAGCT`, ktorého pokrytie fragmentami je však značne nerovnomerné. Ak vezmeme do úvahy aj násobnosť jednotlivých fragmentov, pravdepodobne by sme chceli uprednostniť dlhšie nadslovo `TTAATATTAATATTAATATTAATATTAGCT`, alebo dokonca dvojchromozómovú hypotézu `TTAATATTA` a `ATATTAGCT`, kde je pokrytie fragmentami podstatne rovnomernejšie.

- *Spárované fragmenty* (*pair-end reads*) robia síce formuláciu problému zložitejšou, no dávajú nám množstvo informácie, na základe ktorej možno rozlúsknuť niektoré z problémov uvedených vyššie. Jedná sa o fragmenty, ktoré sú osekvenované z dvoch koncov toho istého segmentu so známou dĺžkou a sú takýmto spôsobom označené (viď obrázok 1.4). Takéto páry, pokiaľ sú dostatočne vzdialené, nám môžu pomôcť určiť správnu dĺžku repetitívnej sekvencie, či pomôcť pri spájaní fragmentov v oblastiach s nízkym pokrytím.

Sformulovať problém zostavovania genómov takým spôsobom, aby zahŕňal všetky faktory, ktoré sme spomenuli na konci predchádzajúcej kapitoly, je veľmi ťažké. Preto tvorcovia softvéru pre zostavovanie genómov, ako napríklad ARACHNE, PHUSION, alebo PCAP, pristúpili k heuristickým riešeniam. Typický softvér na zostavovanie genómu funguje v troch fázach, ktoré obvykle nazývame *overlap*, *layout* a *consensus*.

Overlap. V tejto fáze sa snažíme identifikovať navzájom sa prekrývajúce fragmenty, ktoré potom zostavíme do dlhších sekvencií, ktoré nazývame *kontigy* (contigs).

Úloha však nie je taká jednoduchá, ako sa zdá. Ak by sme totiž chceli porovnať každé dva fragmenty, potrebovali by sme na to čas $O(n^2k)$, kde n je počet fragmentov a $O(k)$

vyjadruje čas, ktorý potrebujeme na porovnanie dvoch fragmentov. Nie je neobvyklé, že n je rádovo niekoľko miliónov: napríklad súčasná verzia genómu potkana je založená na sekvenciách cca. 30 miliónov fragmentov. Pre takéto veľké množstvá fragmentov je kvadratický algoritmus príliš pomalý.

Efektívnejšie algoritmy sú založené na predpoklade, že dva fragmenty, ktoré sa prekrývajú a majú byť zostavené do toho istého kontigu, sú si aj napriek možným chybám sekvenovania veľmi podobné. Preto môžeme predpokladať, že takéto dva fragmenty budú obsahovať 100% zhodnú sekvenciu o dĺžke aspoň k (v praxi sa používa $k \approx 24$, čo pri bežných dĺžkach genómov zaistí, že dva nesúvisiace fragmenty by sme na základe tohto kritéria spojili iba s veľmi malou pravdepodobnosťou).[1] Vo všeobecnosti je potom postup hľadania fragmentov patriacich do toho istého kontigu nasledovný:

1. Vybudujeme zoznam všetkých k-tic zo všetkých fragmentov. Pritom si pre každú k-ticu zapamätáme identifikátor fragmentu, z ktorého táto k-tica pochádza.

2. Zotriedime tento zoznam k-tic podľa abecedy.

3. Skupiny výrazne prekrývajúcich sa fragmentov sa v takomto zozname budú vyskytovať pohromade; na základe informácie o prekryvoch potom môžeme zostavovať kontigy.

Napríklad pre fragmenty 1: `TAATAT`, 2: `GTCTGA`, 3: `TATAAA` a $k = 3$ získame vyššieuvedeným postupom nasledujúci zotriedený zoznam trojíc:

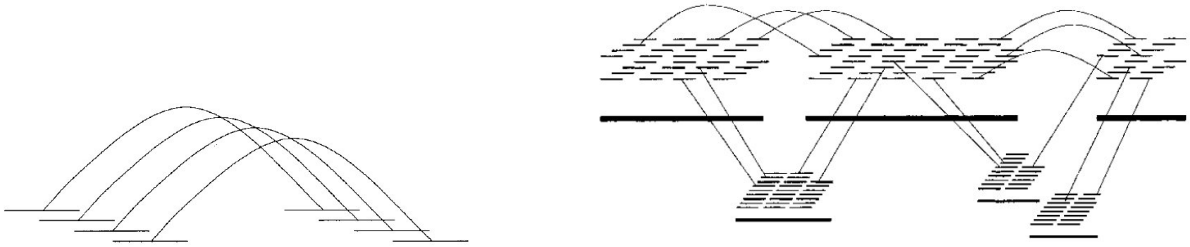
(`AAA`,3) (`AAT`,1) **(`ATA`,1) (`ATA`,3)** (`CTG`,2) (`GTC`,2)
(`TAA`,1) (`TAA`,3) **(`TAT`,1) (`TAT`,3)** (`TCT`,2) (`TGA`,2)

Môžeme teda usúdiť, že fragmenty 1 a 3 treba na základe prekryvov poskladať do kontigu, kým fragment 2 sa neprekrýva dostatočne s inými fragmentami.

Presná implementácia takéhoto postupu sa u jednotlivých programov líši, na vytvorenie efektívneho algoritmu je potrebné navrhnúť vhodné dátové štruktúry a úspornú reprezentáciu v operačnej pamäti počítača, ako aj postup pri samotnom zostavovaní kontigov z jednotlivých fragmentov.

Navyše, nie všetky fragmenty, ktoré takýto algoritmus označí ako podobné, musia byť jednoduchým spôsobom zostaviteľné do toho istého kontigu. Extrémnym príkladom sú opakujúce sa sekvencie, kde genóm obsahuje tesne za sebou niekoľko takmer totožných sekvencií. V takom prípade medzi osekvenovanými fragmentami nájdeme jednak také, ktoré obsahujú koniec jednej kópie a začiatok ďalšej kópie toho istého úseku, ale aj také, ktoré obsahujú koniec poslednej kópie a kúsok sekvencie, ktorá nasleduje bezprostredne za opakujúcim sa úsekom. Pri zostavovaní kontigov tak budeme mať fragmenty, ktoré sa veľmi dobre zhodujú, ale len po určité miesto; za touto hranicou budú fragmenty zrazu veľmi odlišné a nebude sa z nich dať zostaviť jeden kontig. Takéto problémy vyriešime jednoducho tak, že kontig nebudeme rozširovať ďalej za takúto hranicu, zasadenie takéhoto kontigu do kontextu zaistíme v ďalšom kroku.

[1] Podobný predpoklad sa používa pri rýchlom zarovnávaní sekvencií; o tomto probléme budeme viac diskutovať v nasledujúcej kapitole, ktorá sa zarovnávaniu venuje.



Obr. 1.5: **Stratégie fázy layout v programe ARACHNE.** (vľavo) Informácia zo spárovaných fragmentov určuje relatívnu polohu a vzdialenosť kontigov. (vpravo) Sekvenovacie diery a nejednoznačne zostaviteľnú opakujúcu sa sekvenciu je možné vyplniť za pomoci spárovaných fragmentov, ak aspoň na jednej strane je spoľahlivo zostavený kontig. Zdroj: Batzoglou et al. (2002)

Layout. Úlohou tejto fázy je určiť relatívnu polohu a orientáciu jednotlivých kontigov, ako aj ich približné vzdialenosti. Kontigy na seba nemusia úplne nadväzovať z viacerých dôvodov. Jednak niektoré časti genómu nemusia byť pokryté osekvenovanými fragmentami. Je jasné, že v týchto miestach sekvenciu nezískame, no chceli by sme aspoň okolité kontigy správne zoradiť a odhadnúť veľkosť takýchto *sekvenovacích dier* (sequencing gaps). V praxi potom sekvenovacie diery v rámci dlhšej sekvencie reprezentujeme dlhou postupnosťou znakov `N` (neznámy nukleotid). Ďalšou príčinou rozbitia sekvencie na viacero kontigov môžu byť opakujúce sa sekvencie a ďalšie nejednoznačnosti, ktoré nebolo možné vyriešiť v prechádzajúcej fáze. Výsledné bloky kontigov sa potom nazývajú *superkontigy*.

V tejto fáze využijeme informáciu, ktorá je obsiahnutá v spárovaných fragmentoch (pair-end reads). Keďže spárované fragmenty sú osekvenované z koncov toho istého segmentu DNA približne známej dĺžky, vieme ako ďaleko a v akej orientácii by tieto fragmenty mali byť vo výslednej sekvencii uložené. Spárované fragmenty tak vytvárajú medzi jednotlivými kontigmi väzby, ktoré určujú ich vzájomné vzdialenosti, ako aj orientáciu. Takisto pomocou nich dokážeme vytvoriť heuristiky, ktoré dokážu aspoň čiastočne zostaviť opakujúce sa sekvencie. Obrázok 1.5 demonštruje postupy, ktorými možno využiť takúto informáciu na rozšírenie a spájanie kontigov.

Consensus. Každá pozícia v superkontigu je potenciálne pokrytá niekoľkými osekvenovanými fragmentami, pričom na danej pozícii sa fragmenty môžu aj líšiť, či už z dôvodu sekvenovacej chyby, alebo výskytu polymorfizmu. V tejto fáze vytvoríme finálne sekvencie superkontigov, pričom v prípade nezhodujúcich sa nukleotidov použijeme nejaký typ väčšinového pravidla, berúc do úvahy aj kvalitu jednotlivých báz vo fragmentoch.

Prístup overlap-layout-consensus sa stal štandardom pri zostavovaní sekvencií z dlhých fragmentov (ako napríklad tých, ktoré získame pomocou Sangerovho sekvenovania). Kvalita výsledku v tomto prípade závisí nielen od algoritmu na zostavovanie, ale aj od množstva sekvencie, ktoré máme k dispozícii. Množstvo sekvencie spravidla meriame násobkom dĺžky genómu, resp. hĺbkou pokrytia; pri 5× pokrytí bude každé miesto genómu v priemere pokryté piatimi fragmentami, no keďže sekvenovanie je náhodný proces, pokrytie na konkrétnych miestach bude značne nerovnomerné a výsledok môže obsahovať veľké množstvo sekvenovacích dier. Tabuľka 1.2 ukazuje úspešnosť softvéru ARACHNE pri rôz-

Tabuľka 1.2: **Úspešnosť zostavovania genómu v závislosti od pokrytia.** Z genómu *D. melanogaster* bolo simuláciou vytvorených príslušné množstvo fragmentov. Tieto boli následne zostavované pomocou programu ARACHNE a porovnávané s pôvodnou sekvenciou. Podľa Batzoglou et al. (2002).

	Počet	Priem. dĺžka	Chýb
10× pokrytie:			
Kontigy	678	174 kB	
Superkontigy	65	1.8 MB	115
5× pokrytie:			
Kontigy	8774	13 kB	
Superkontigy	450	254 kB	175

nych pokrytiach genómu vínnej mušky *D. melanogaster*; ako vidíme, celistvosť zostavenia genómu od hĺbky pokrytia značne závisí. Pri Sangerovom sekvenovaní sa za draft genómu považuje 2× pokrytie (ktoré bude mať značné množstvo sekvenovacích dier a veľmi veľa kontigov), kým kvalitne osekvenovaný genóm bude mať pokrytie minimálne 7×.

1.3 De Bruijnove grafy

Ako sme naznačili v predchádzajúcich kapitolách, zostavovanie genómov je vo všeobecnosti veľmi ťažký problém, ktorý možno v praxi riešiť iba heuristickými metódami. Čo i len sformulovať problém tak, aby zahrnul všetku informáciu obsiahnutú v dátach je veľmi ťažké.

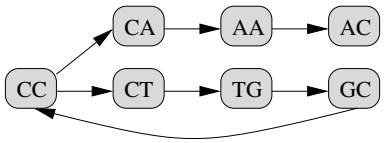
Prístup, ktorý si opíšeme v tejto kapitole, vychádza z toho, že pri sekvenovaní je obvykle dát veľmi veľa, a preto si môžeme dovoliť časť informácie v nich obsiahnutej ignorovať. Odmenou bude jasne formulovaný problém, ktorý možno dokonca ľahko a efektívne riešiť. Samozrejme, nevýhodou je, že sa môže stať, že výsledné riešenie bude v rozpore so vstupnými dátami, teda s tou časťou, ktorú sme sa rozhodli nevyužiť.

Pre jednoduchosť budeme v tejto kapitole predpokladať, že všetky fragmenty prichádzajú z jedného vlákna (tzn. všetky sú orientované v jednom smere), že sa snažíme zostaviť jeden chromozóm, ktorý je navyše úplne pokrytý fragmentami (žiadne sekvenovacie diery). Aj keď sú tieto predpoklady nerealistické, na ilustráciu princípu to postačí.

Prvým krokom algoritmu je všetky fragmenty nasekať na ešte menšie prekrývajúce sa fragmenty, každý presne dĺžky k (tieto nové fragmenty budeme nazývať k-tice). Ak má teda fragment pôvodne dĺžku 20 a $k = 10$, vytvoríme z neho 11 k-tic, každá dĺžky presne 10, pričom jednotlivé k-tice začínajú na pozíciách 1, 2, ..., 11. Z takýchto k-tic vytvoríme tzv. *de Bruijnov graf*, ktorý je definovaný nasledovne:

- Vrcholmi de Bruijnovho grafu budú všetky kúsky dĺžky $k-1$, ktoré sa vyskytujú v našich dátach.

- Jednotlivé k-tice budú reprezentované orientovanými hranami de Bruijnovho grafu, pričom k-tica $s_1 s_2 \ldots s_k$ spája vrcholy $s_1 s_2 \ldots s_{k-1}$ a $s_2 s_3 \ldots s_k$.



Obr. 1.6: **Príklad de Bruijnovho grafu.** Fragmenty `CCTGCC`, `GCCAAC`, $k = 3$.

Príklad de Bruijnovho grafu je na obrázku 1.6. V takomto grafe genómu zodpovedá ťah, ktorý používa všetky hrany de Bruijnovho grafu. Napríklad genómu `CCTGCCAAC` v našom príklade zodpovedá ťah (`CC`),(`CT`),(`TG`),(`GC`),(`CC`),(`CA`),(`AA`),(`AC`). Ak navyše predpokladáme, že k je dostatočne veľké, aby sa v genóme nevyskytli dve k-tice, ktoré sú úplne zhodné, tak každá hrana bude použitá práve raz. Cesta v orientovanom grafe, ktorá používa každú hranu grafu práve raz sa nazýva *Eulerovský ťah* a známy výsledok z teórie grafov nám hovorí, kedy graf takýto Eulerovský ťah obsahuje.

Veta 1. Orientovaný graf má Eulerovský ťah z vrcholu u do vrcholu v práve vtedy, keď po pridaní hrany (v, u) je graf silne súvislý a v každom vrchole sa počet vchádzajúcich hrán rovná počtu vychádzajúcich hrán.

Navyše, ak je táto podmienka splnená, Eulerovský ťah možno v grafe nájsť veľmi jednoducho v čase $O(n + m)$, kde n je počet vrcholov a m je počet hrán.

Ak teda zostavíme z našich fragmentov de Bruijnov graf a zistíme, že tento graf má Eulerovský ťah, zdalo by sa, že genóm môžeme zostaviť vskutku jednoducho: stačí zobrať sekvenciu zodpovedajúcu prvému vrcholu v tomto ťahu a pri každom prechode hrany do nového vrcholu pridať nukleotid zodpovedajúci poslednému nukleotidu tohto nového vrcholu. Samozrejme, problémom môže byť fakt, že pri tvorbe de Bruijnovho grafu neuvažujeme niektoré závislosti v pôvodných fragmentoch. V našom príklade máme napríklad zaručené, že vo výslednom genóme bude obsiahnutá sekvencia `GCC`, avšak nemôžeme garantovať, že v ňom bude obsiahnutá aj sekvencia `GCCA`, keďže pri tvorbe de Bruijnovho grafu sme rozsekali pôvodný fragment `GCCAAC` na trojice. Výsledný ťah tak môže byť nekonzistentný s pôvodnými fragmentami.

V praxi samozrejme veľmi často narazíme na problém, že výsledný de Bruijnov graf nemá Eulerovský ťah, alebo naopak má takých ťahov viacero. Preto je potrebné túto metódu doplniť množstvom heuristík, ako napríklad:

- Odstraňovanie vrcholov a hrán, ktoré z rôznych dôvodov môžu zodpovedať sekvenovacím chybám.
- Znásobovanie hrán, ak si myslíme, že k-tica sa v genóme vyskytuje viackrát.
- Postupné vyčleňovanie podciest zodpovedajúcich pôvodne zadaným dlhším fragmentom a ich nahradenie jedinou hranou.
- Použitie spárovaných fragmentov: ak v grafe existuje jediná cesta vhodnej dĺžky, ktorá spája vrcholy zodpovedajúce spárovaným fragmentom, zmeníme ju na jedinú hranu.

Tabuľka 1.3: Prehľad sekvenovacích technológií druhej generácie

	454 (Roche)	Illumina (Solexa)	SOLiD (AB)
Čas/beh	10 h	2–5 dní	6 dní
Dáta/beh	400 Mb	3000 Mb	4000 Mb
Dĺžka fragmentu	400bp	35–100bp	35–50bp
Cena na beh	$8500	$9 000	$17 500
Cena na Mb	$85	$6	$6

Pre úspech celej metódy je dôležitý aj výber vhodného k. Na jednej strane, príliš malé k zavedie do grafu príliš veľa nejednoznačnosti vo forme opakujúcich sa k-tic. Ak aj k zväčšujeme, opakujúcich sa k-tic sa obvykle úplne nezbavíme, keďže genómy obsahujú pomerne dlhé opakujúce sa sekvencie. Navyše, príliš veľké k znamená veľké nároky na operačnú pamäť počítača, a samozrejme musíme vyradiť všetky fragmenty, ktorých dĺžka je menšia ako $k + 1$.

De Bruijnove grafy predstavujú metódu, ktorá zaznamenala úspech najmä u malých genómov a taktiež pri aplikácii metód druhej generácie, ktoré produkujú veľké množstvo krátkych fragmentov. V takom prípade totiž voľbou menšieho k nestrácame príliš mnoho informácie, a obrovské množstvá dát vyžadujú rýchle spracovanie pomocou efektívnych algoritmov.

1.4 Sekvenovacie technológie novej generácie

Postupný vývoj Sangerovho sekvenovania priniesol spoľahlivú technológiu schopnú produkovať fragmenty dlhé až 1000 bp. V oblasti sekvenovania však nastal prudký rozvoj, keď sa v roku 2005 objavili komerčne dostupné sekvenátory druhej generácie (high-throuput second generation sequencers). Aj keď technológia sa u rôznych platforiem líši, sekvenátory druhej generácie majú niekoľko spoločných čŕt:

- Sekvenovanie obrovského množstva fragmentov paralelne z tej istej vzorky.
- Rýchlejšie a oveľa lacnejšie sekvenovanie.
- Odbúranie náročného procesu klonovania.
- Oveľa kratšie fragmenty (od 35 do 400 báz podľa použitej technológie).

Prehľad parametrov sekvenátorov druhej generácie je v tabuľke 1.3. Príchod nových technológií a zníženie ceny podnietil obrovskú expanziu sekvenovania. Kým pred príchodom sekvenátorov druhej generácie sa sekvenovanie sústreďovalo takmer výlučne vo veľkých genómových centrách, v súčasnosti si sekvenovacie technológie môže dovoliť prakticky každé väčšie výskumné laboratórium.

Príchod ďalšej generácie sekvenovania tiež inšpiroval rozvoj v oblasti bioinformatiky. Zostavovanie genómov sa stalo ešte náročnejším, keďže nové technológie produkujú oveľa kratšie fragmenty, zato v oveľa väčších množstvách. Nové aplikácie sekvenovania, ktoré

by sa predtým javili ako príliš finančne náročné, priniesli nové bioinformatické problémy. Množstvo dát v súčasnosti rastie podstatne rýchlejšie ako Moorov zákon, čo znamená ďalšie výzvy v oblasti uchovávania, prenosu, aj spracovania dát.

Samozrejme, druhou generáciou vývoj sekvenovania nekončí. Vývoj nových platforiem pokračuje dvoma smermi: znižovanie množstva DNA, ktoré je potrebné na sekvenovanie (až na úroveň jedinej molekuly) a sekvenovanie superdlhých fragmentov (umožňujúcich odbúrať množstvo problémov so zostavovaním genómov).

1.5 Ďalšie aplikácie sekvenovania

Okrem sekvenovania nových organizmov sú v súčasnosti sekvenovacie technológie využívané na množstvo ďalších účelov.

Populačná genetika. V tomto prípade sekvenujeme ďalších jedincov druhu, ktorý sme už predtým osekvenovali a u ktorého už máme zostavený tzv. referenčný genóm. Obvykle sekvenujeme množstvo krátkych fragmentov, ktoré potom mapujeme na referenčný genóm, pričom sa zaujímame hlavne o rozdiely medzi referenčným genómom a sekvenciou daného jedinca. Sekvenovanie tu nahradilo staršiu techniku genotypovania pomocou génových čipov (genotyping microarrays). Technológia nám umožňuje zodpovedať otázky o tom, ako jednoduché genetické rozdiely ovplyvňujú fenotyp, ale aj otázky o populačnej štruktúre a histórii druhu. Prístupnosť sekvenovania podnietila projekt sekvenovania 1000 ľudských genómov a rozprúdila úvahy o možnostiach personalizovanej medicíny v dohľadnej budúcnosti.

Metagenomika. Metagenomika alebo environmentálne sekvenovanie sa snaží zodpovedať na otázku, aké skupiny organizmov žijú v rôznych ekosystémoch (morská voda, extrémne prostredia), v našich telách (črevná a žalúdočná flóra, ústna dutina, koža) a pod. Na rozdiel od bežného sekvenovania sa nesnažíme izolovať jednotlivé organizmy, keďže ide o veľké množstvo druhov žijúcich vo vzájomnej symbióze, o ktorých navyše nevieme dosť, aby sme ich vedeli kultivovať. Namiesto toho sekvenujeme jednoducho zmes DNA z rôznych organizmov, každý fragment tak môže pochádzať z iného organizmu. Výsledkom bývajú krátke kontigy, na základe ktorých sa snažíme pomocou štatistických metód odhadnúť parametre diverzity v metagenomickej vzorke, prípadne sa snažíme objaviť nové gény a mutácie, ktoré sa nevyskytujú v dobre preštudovaných organizmoch.

Hľadanie génov a meranie transkripcie. Namiesto toho, aby sme sekvenovali DNA, môžeme sekvenovať aj RNA (technológia sa nazýva RNA-Seq). Zarovnaním osekvenovaných fragmentov RNA k referenčnému genómu získame obraz o tom, ktoré časti genómu sa v živej bunke prepisujú, a tým aj predstavu o polohe jednotlivých génov. Navyše môžeme porovnávať rozdiely v transkripcii za rôznych podmienok, čo nám pomáha odhadnúť funkciu jednotlivých oblastí genómu. Sekvenovanie v tomto prípade nahrádza staršiu technológiu merania a porovnávanie transkripcie na základe génových čipov (microarrays).

Väzobné miesta transkripčných faktorov. V živej bunke sa na DNA viažu rôzne proteíny, čím sa menia jej lokálne vlastnosti a pomáhajú alebo zabraňujú transkripcii

určitých častí DNA do RNA. Pomocou tzv. imunoprecipitácie je možné pre konkrétny transkripčný faktor vyselektovať len tie kúsky genómu, na ktoré sa tento transkripčný faktor viaže. Následné osekvenovanie týchto kúskov a ich namapovanie na referenčný genóm nám umožňuje získať prehľad o väzobných miestach tohto transkripčného faktoru. Táto technológia (ChIP-seq) nahradila v súčasnosti podobnú technológiu založenú na použití génových čipov (ChIP-chip). Podobným spôsobom možno sledovať nielen transkripčné faktory, ale aj iné špeciálne miesta v DNA (napr. chromatínová štruktúra, lokalizácia v blízkosti membrán a pod.).

1.6 Zhrnutie

V tejto kapitole sme sa zaoberali sekvenovaním genómov a problémom zostavovania genómov z kratších fragmentov. Sekvenovanie genómov je zložitý proces, v rámci ktorého získame veľké množstvo malých fragmentov DNA sekvencie. Tieto fragmenty je potom potrebné bioinformatickými metódami poskladať do dlhších sekvencií na základe prekryvov. Základná metóda sa nazýva overlap–layout–consensus.

V súčasnosti sú k dispozícii tzv. technológie novej generácie, ktoré za prijateľnú cenu vedia z jednej vzorky osekvenovať veľké množstvo pomerne krátkych fragmentov. Veľké množstvá dát, ako aj krátke fragmenty kladú veľké nároky na bioinformatické metódy. Pre tento typ dát sa na zostavovanie najčastejšie používajú tzv. de Bruijnove grafy.

V rámci sekvenovania, ako aj pri zostavovaní genómu môže dochádzať k rôznym chybám. Sekvenovacie chyby sú často doprevádzané nízkym skóre kvality. V každom prípade sa ale v zostavených sekvenciách vyskytujú chyby, sekvenovacie diery, v niektorých prípadoch môžu byť úseky zostavenej sekvencie chybne usporiadané, prípadne sekvenciu nemožno zostaviť až na úroveň chromozómov, ale len tzv. superkontigov. S týmito chybami je nutné rátať pri tvorbe záverov a pri dizajnovaní ďalších experimentov.

Sekvenovanie sa v súčasnosti používa aj na množstvo ďalších účelov, ako napr. sekvenovanie jedincov v populačnej genetike, metagenomika, RNA-Seq a ChIP-Seq. V budúcnosti je pravdepodobné, že DNA sekvencia konkrétneho človeka bude súčasťou jeho zdravotnej dokumentácie a umožní lepšie cielenie liečby rôznych ochorení.

Kapitola 2

Zarovnávanie sekvencií

V tejto kapitole sa budeme venovať jednému zo základných problémov bioinformatiky, a to hľadaniu podobností medzi sekvenciami DNA alebo proteínov. Vďaka dostupnosti sekvenovacích technológií máme k dispozícii veľké databázy sekvencií a môžeme hľadať, či sa niektoré sekvencie alebo ich časti zhodujú alebo aspoň podobajú. Podobnosti medzi sekvenciami v databázach pochádzajú buď z toho, že sa viackrát sekvenovala tá istá sekvencia, alebo že sa dve sekvencie vyvinuli mutáciami zo sekvencie v spoločnom predkovi. Takéto sekvencie sa nazývajú *homologické* a pod pojmom *hľadanie homológov* (*homology search*) sa myslí hľadanie takých podobností medzi sekvenciami, ktoré s veľkou pravdepodobnosťou vznikli práve zdielanou evolučnou históriou.

Nájdené podobnosti väčšinou zobrazujeme vo forme *zarovnania* (*sequence alignment*) ako na obrázku 2.1. Každý riadok zarovnania je jedna zo sekvencií, pričom do sekvencií pridáme podľa potreby pomlčky tak, aby boli oba riadky rovnako dlhé a aby v stĺpcoch pod sebou boli čo najčastejšie rovnaké bázy. Skupinu susedných pomlčiek v zarovnaní nazývame *medzera* (*gap*). Dva základné problémy zarovnávania sekvencií sú lokálne a globálne zarovnanie. Pri *globálnom zarovnaní* dvoch sekvencií zarovnávame tieto sekvencie celé a pri *lokálnom zarovnaní* hľadáme v sekvenciách čo najpodobnejšie oblasti a tie zarovnáme (obr. 2.1). Ako uvidíme nižšie, pomocou jednoduchého skórovania je možné obidva problémy formulovať ako presne definovaný informatický problém.

Sekvencia 1:

```
acgcctccacccccgccctactcgggcagtttaac
ccttgttgttcacttgcagacatcgtgaacacggcc
cggCCCGACGAGAAGGCCATAATGACCTATGTGTCC
AGCTTCTACCATGCCTTTtcaggagcgcagaaggta
ccgagcagggccaggcaggccctcctcgccgccacc
```

Sekvencia 2:

```
tgatgccgaggatgtgttcgtcgagcatCCGGACGA
GAAGTCCATCACCTACGTGGTCACCTACTATCACTA
ACTTTgcaaactcaagcaggagacggtgcagggcat
aagcgtatcggtaaggtggtcggcattgccatggag
aacgacaaaatggtccacgactacgagaacttcaca
```

Lokálne zarovnanie:

```
CCCGACGAGAAGGCCATAATGACCTATGTGTCCAGCTTCTACCA-TGCCTTT
CCGGACGAGAAGTCCAT---CACCTACGTGGTCACCTACTATCACTAACTTT
```

Obr. 2.1: Dve sekvencie a ich lokálne zarovnanie. Zarovnaná časť sekvencií je zvýraznená farebne a veľkými písmenami.

2.1 Použitie zarovnávania

Ukážme si teraz niekoľko príkladov situácií, v ktorých sa typicky používajú programy na zarovnávanie sekvencií. Najjednoduchším príkladom je *mapovanie*: napríklad osekvenujeme RNA extrahovanú zo vzorky buniek a chceme vedieť, kde v genóme sa nachádza gén, ktorého transkripciou táto RNA vznikla. Zarovnávame teda sekvenciu RNA so sekvenciou genómu. Očakávame buď presnú zhodu, alebo len malý počet rozdielov, ktoré zodpovedajú sekvenovacím chybám alebo polymorfizmom (rozdielom medzi jedincami). Ako sme videli v kapitole 1, podobne na genóm mapujeme aj segmenty osekvenované z genomickej DNA, napríklad pri porovnávaní genómov rôznych jedincov v rámci druhu.

Zaujímavejší je však prípad, keď pomocou zarovnaní hľadáme homológy, t.j. sekvencie, ktoré sa vyvinuli mutáciami z toho istého predka. V takom prípade zarovnanie predstavuje hypotézu o evolúcii týchto sekvencií, kde predpokladáme, že zarovnané bázy pochádzajú z rovnakej bázy v spoločnom predkovi, pričom mohlo prípadne dôjsť k mutáciám. Bázy zarovnané s pomlčkami zodpovedajú oblastiam, ktoré boli buď vložené v jednej zo zarovnaných sekvencií, alebo zmazané v druhej (len zo zarovnania dvoch sekvencií nevieme navzájom odlíšiť inzercie a delécie). V tejto kapitole si ukážeme, ako pomocou štatistických modelov odhadnúť, či zarovnanie dvoch sekvencií predstavuje skutočnú homológiu, alebo či ide o dve sekvencie, ktoré sa navzájom podobajú iba náhodou. Ide však iba o hypotézu, ktorú nie je možné priamo experimentálne overiť, nakoľko väčšinou nemáme ako zistiť skutočnú históriu porovnávaných sekvencií.

Hľadanie homológov je užitočné aj na skúmanie samotnej evolúcie, ako uvidíme v kapitolách 3 a 5. V zarovnaných sekvenciách môžeme napríklad skúmať, ako často sa mení jedna báza na inú a či sa frekvencie mutácií líšia napríklad medzi génmi a nekódujúcimi oblasťami. Ak poznáme homologické sekvencie z viacerých biologických druhov, môžeme sa snažiť odvodiť ich evolučnú históriu vo forme evolučného stromu. Na to väčšinou potrebujeme najskôr vytvoriť *viacnásobné zarovnanie* (*multiple sequence alignment*), v ktorom máme viacero riadkov, každý zodpovedajúci jednej sekvencii (obr. 2.11). Stĺpce, rovnako ako v zarovnaní dvoch sekvencií, zodpovedajú pozíciám, ktoré pochádzajú z tej istej bázy v sekvencii spoločného predka.

Nájdené homológy tiež často používame na usudzovanie o funkcii skúmaných sekvencií. Ako sa počas evolúcie postupne mutáciami mení sekvencia, môže dôjsť aj k menším zmenám v jej funkcii. Vďaka prírodnému výberu však k výrazným zmenám funkcie dochádza menej často, najmä pri génoch, ktoré sú obzvlášť dôležité pre prežitie. Takže ak zistíme, že naša skúmaná sekvencia sa podobá na nejakú sekvenciu z databázy, ktorej funkcia je známa, môžeme vysloviť hypotézu, že aj naša funkcia má tú istú alebo podobnú funkciu. Táto hypotéza však nie je vždy správna. Treba mať na pamäti, že podobnosť sekvencií, homológia a podobnosť funkcií sú vzťahy, ktoré síce spolu súvisia, ale nie sú totožné.

Treba tiež vždy dobre zvážiť, či pre daný problém chceme hľadať lokálne alebo globálne zarovnanie. Ak máme hypotézu, že skúmané sekvencie sú homologické v celej svojej dĺžke, môžeme ich zarovnávať globálne. To sa najčastejšie uplatní pri evolučne príbuzných proteínových sekvenciách. Pri vyhľadávaní podobností medzi skúmanou sekvenciou a veľkou dabázou sekvencií sa môže stať, že iba určitá časť našej sekvencie bude podobná so známou sekvenciou. Preto pri hľadaní v databázach väčšinou hľadáme lokálne zarovnania. Lokálne zarovnania sa tiež používajú pri porovnávaní celých genómov, lebo v

C	A	C	T	C		C	-	A	C	T	C
C	G	A	C	T		C	G	A	C	T	-
+1	−1	−1	−1	−1		+1	−1	+1	+1	+1	−1
Skóre −3						Skóre +2					

Obr. 2.2: Príklad dvoch zarovnaní sekvencií `CACTC` a `CGACT` s rôznym skóre.

evolúcii môže dôjsť k zmene poradia väčších oblastí v rámci chromozómov alebo k presunom medzi chromozómami, takže nie je možné navzájom zarovnať do jedného zarovnania celé chromozómy.

2.2 Definícia problému lokálneho a globálneho zarovnania

Ak chceme vytvoriť algoritmy a softvérové nástroje na zarovnávanie sekvencií, potrebujeme presne definovať problém, ktorý majú riešiť. Ak vezmeme dve sekvencie, existuje veľa spôsobov, ako do nich môžeme povkladať pomlčky a musíme definovať, ktoré z týchto zarovnaní je najlepšie. Pripomíname, že nestačí stanoviť len biologický cieľ (chceme dať pomlčky tak, aby boli pod sebou homologické bázy), ale treba ho úplne presne matematicky definovať.

Vytvoríme teda jednoduchý skórovací systém, v ktorom každému zarovnaniu vieme určiť skóre a za najlepšie budeme považovať zarovnanie s najvyšším skóre. V našom systéme určíme skóre pre každý stĺpec zarovnania a skóre celého zarovnania bude súčtom skóre jednotlivých stĺpcov. Ak stĺpec obsahuje dve rovnaké bázy, jeho skóre bude +1, ak obsahuje dve rôzne bázy, jeho skóre bude −1 a ak obsahuje pomlčku a bázu jeho skóre bude tiež −1 (obr. 2.2).

S využitím takéhoto jednoduchého systému skórovania môžeme zadefinovať dve základné formy problému zarovnávania sekvencií.

Definícia 2 (Problém globálneho zarovnania (global alignment problem))**.** Vstupom sú dve sekvencie $X = x_1x_2 \dots x_n$ a $Y = y_1y_2 \dots y_m$. Výstupom je zarovnanie X a Y s najvyšším skóre.

Definícia 3 (Problém lokálneho zarovnania (local alignment problem))**.** Vstupom sú dve sekvencie $X = x_1x_2 \dots x_n$ a $Y = y_1y_2 \dots y_m$. Výstupom je zarovnanie nejakých podreťazcov $x_i \dots x_j$ a $y_k \dots y_\ell$ s najvyšším skóre.

Toto sú už dobre definované informatické problémy. Nižšie si ukážeme známe algoritmy, ktoré riešia tento problém, t.j. pre vstupné sekvencie nájdu globálne alebo lokálne zarovnanie s najvyšším možným skóre. Toto zarovnanie bude predstavovať najlepšieho kandidáta na potenciálnu homologickú sekvenciu.

2.3 Skórovanie zarovnaní

Definícia problému lokálneho a globálneho zarovnania vyžaduje, aby sme mali pevne stanovený systém na skórovanie zarovnaní, pričom doteraz sme zhodám priradili skóre

```
.  A  R  N  D  C  Q  E  G  H  I  L ...
A  4 -1 -2 -2  0 -1 -1  0 -2 -1 -1
R -1  5  0 -2 -3  1  0 -2  0 -3 -2
N -2  0  6  1 -3  0  0  0  1 -3 -3
D -2 -2  1  6 -3  0  2 -1 -1 -3 -4
C  0 -3 -3 -3  9 -3 -4 -3 -3 -1 -1
Q -1  1  0  0 -3  5  2 -2  0 -3 -2
E -1  0  0  2 -4  2  5 -2  0 -3 -3
G  0 -2  0 -1 -3 -2 -2  6 -2 -4 -4
H -2  0  1 -1 -3  0  0 -2  8 -3 -3
I -1 -3 -3 -3 -1 -3 -3 -4 -3  4  2
L -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4
...
```

Obr. 2.3: Úryvok z matice BLOSUM62 na skórovanie proteínových zarovnaní (Henikoff and Henikoff, 1992). Celá matica má 20 riadkov a 20 stĺpcov.

+1 a nezhodám a medzerám −1. Tieto hodnoty si však môžeme zvoliť aj inak, napríklad zhodám môžeme priradiť skóre +2 a nezhody môžeme hodnotiť inak ako medzery. Našim cieľom je zaviesť taký systém skórovania zarovnaní, aby zarovnania, ktoré zodpovedajú skutočným evolučným vzťahom medzi homologickými sekvenciami mali pokiaľ možno vysoké skóre a zarovnania medzi sekvenciami, ktoré nie sú evolučne príbuzné, mali nízke skóre. V tejto časti si niečo povieme o zostavovaní takýchto skórovacích systémov.

2.3.1 Skórovacie matice

Nie vždy musíme všetky zhody alebo všetky nezhody skórovať rovnako, ale ich skóre môže závisieť aj od toho, aké konkrétne bázy sú v danom stĺpci zarovnania. Použijeme teda *skórovaciu maticu*, ktorá určuje skóre pre každú možnú dvojicu báz alebo aminokyselín. Takéto matice sa používajú najmä pri zarovnávaní proteínov, nakoľko niektoré dvojice aminokyselín majú podobné chemické vlastnosti a preto by sme ich zámenu chceli penalizovať menej ako zámenu výrazne odlišných aminokyselín. Napríklad na obrázku 2.3 vidíme úryvok z matice BLOSUM62, ktorá sa často používa na skórovanie proteínových zarovnaní. Napríklad izoleucín (`I`) a leucín (`L`) sú pomerne podobné hydrofóbne aminokyseliny a zmena z jednej na druhú má kladné skóre +2. Naopak zmena z leucínu na hydrofilnú kyselinu asparágovú (`D`) má záporné skóre −4.

Pre 20 rôznych aminokyselín je ťažké ručne vymyslieť spôsob, ako skórovať všetky možné dvojice, ktoré sa v zarovnaní môžu vyskytnúť. Preto si ukážeme, ako skórovaciu maticu odvodiť systematicky pomocou pravdepodobnostných modelov. Budeme na to potrebovať sadu vzorových zarovnaní, ktoré zodpovedajú skutočným evolučným vzťahom. Takúto sadu zarovnaní budeme nazývať aj *trénovacia množina* (*training set*).

Z trénovacej množiny zarovnaní vezmeme všetky stĺpce bez medzier a spočítame, ako často sa medzi nimi vyskytujú jednotlivé kombinácie báz alebo aminokyselín. Nech $p(x, y)$ je frekvencia, s ktorou sa v zarovnaniach vyskytuje stĺpec obsahujúci bázy x a y. Napríklad $p(\mathtt{A}, \mathtt{A}) = 0.18$ znamená, že zo všetkých našich trénovacích stĺpcov 18% tvoria stĺpce obsahujúce dve bázy `A` pod sebou.

Z týchto pravdepodobností pre jednotlivé stĺpce teraz môžeme vytvoriť jednoduchý pravdepodobnostný model H pre dlhšie zarovnania bez medzier. Tento model každému zarovnaniu určitej pevnej dĺžky n priradí nejakú pravdepodobnosť, pričom chceme, aby

zarovnania, ktoré sa viac podobajú na zarovnania z trénovacej množiny, mali vyššiu pravdepodobnosť a zarovnania, ktoré sa podobajú menej, nižšiu pravdepodobnosť. Uvažujme teda zarovnanie bez medzier dĺžky n obsahujúce sekvencie $X = x_1 \dots x_n$ a $Y = y_1 \dots y_n$. V i-tom stĺpci zarovnania budú bázy x_i a y_i, ktoré majú v našej trénovacej množine pravdepodobnosť $p(x_i, y_i)$. Budeme predpokladať, že jednotlivé stĺpce zarovnania sú navzájom nezávislé a teda tieto pravdepodobnosti jednotlivých stĺpcov môžeme navzájom vynásobiť. Takže pravdepodobnosť zarovnania $X = x_1 \dots x_n$ a $Y = y_1 \dots y_n$ v modeli H je

$$\Pr(X, Y \mid H) = \prod_{i=1}^{n} p(x_i, y_i).$$

Aby sme zadefinovali skórovaciu maticu, potrebujeme zaviesť ešte jeden pravdepodobnostný model, ktorý budeme označovať R. Kým model H zodpovedal skutočným zarovnaniam, model R zodpovedá prípadu, keď spolu zarovnáme úplne nesúvisiace sekvencie X a Y rovnakej dĺžky. Aby sme model R definovali, potrebujeme si v sekvenciách z trénovacej množiny spočítať, ako často sa vyskytujú jednotlivé bázy. Nech $p(x)$ je frekvencia, s ktorou sa vyskytuje báza x. Model R predpokladá, že sekvencie X a Y sú navzájom nezávislé a že navyše každá báza v oboch sekvenciách je nezávislá od iných báz. Zase teda stačí vynásobiť hodnoty $p(x_i)$ a $p(y_i)$ pre všetky pozície i, čím dostaneme vzorec

$$\Pr(X, Y \mid R) = \left(\prod_{i=1}^{n} p(x_i) \right) \left(\prod_{i=1}^{n} p(y_i) \right)$$

Ak vezmeme nejaké sekvencie X a Y dĺžky n a ich zarovnanie bez medzier, vieme spočítať pravdepodobnosť tohto zarovnania v modeloch H aj R. Ak sú sekvencie X a Y homológy, očakávame, že v modeli H zodpovedajúcom skutočným homológom bude mať ich zarovnanie vyššiu pravdepodobnosť ako v modeli R, ktorý zodpovedá nesúvisiacim sekvenciám. Naopak, ak sekvencie X a Y spolu nesúvisia, očakávame, že budú mať väčšiu pravdepodobnosť v modeli R než H. Ako skóre zarovnania si zvolíme logaritmus pomeru týchto dvoch pravdepodoností (*log likelihood ratio* alebo tiež *log odds ratio*)

$$\log \frac{\Pr(X, Y \mid H)}{\Pr(X, Y \mid R)}. \tag{2.1}$$

Pre homologické sekvencie by mal byť teda pomer pravdepodobností väčší ako 1 a celkové skóre kladné. Naopak, pre nesúvisiace sekvencie očakávame pomer pravdepodobností menší ako 1 a celkové skóre záporné.

Aj keď to možno nie je vidno na prvý pohľad, skóre definované výrazom 2.1 je možné rozpísať ako súčet čiastkových skóre pre jednotlivé stĺpce zarovnania sekvencií X a Y:

$$\log \frac{\Pr(X, Y \mid H)}{\Pr(X, Y \mid R)} = \log \frac{\prod_{i=1}^{n} p(x_i, y_i)}{\left(\prod_{i=1}^{n} p(x_i)\right)\left(\prod_{i=1}^{n} p(y_i)\right)} = \sum_{i=1}^{n} \log \frac{p(x_i, y_i)}{p(x_i)p(y_i)}.$$

Skóre pre stĺpec obsahujúci bázu x a bázu y teda je $\log \frac{p(x,y)}{p(x)p(y)}$.

Vo všeobecnosti sú takto získané skóre reálne čísla, býva však zvykom všetky skóre prenásobiť nejakou konštantou, aby sa dostali do vhodného rozpätia hodnôt a potom zaokrúhliť na celé čísla. Násobenie (kladnou) konštantou nezmení nájdené zarovnania,

lebo skóre každého zarovnania sa tiež len prenásobí tou istou konštantou a teda zarovnanie s maximálnym skóre zostane to isté. Zaokrúhľovanie na celé čísla zjednodušuje výpočet, aj keď dochádza k určitej strate presnosti.

Týmto postupom boli vytvorené aj matice BLOSUM (BLOcks of aminoacid SUbstitution Matrix, Henikoff and Henikoff (1992)), ktoré sa často používajú na zarovnávanie proteínov (obr. 2.3). Ako trénovaciu množinu autori vybrali zarovnania proteínov z databázy BLOCKS, pričom napríklad pri tvorbe matice BLOSUM62 použili iba dvojice zarovnaní s najviac 62% zhodami. Matica BLOSUM45 sa tak hodí skôr na hľadanie zarovnaní vzdialenejších homológov, kým matica BLOSUM62 bližších.

Problémom tohto prístupu je voľba trénovacej množiny zarovnaní. Ak na vytvorenie tejto množiny využijeme nejakú už existujúcu skórovaciu maticu, jej voľba ovplyvní získané zarovnania a tým aj novú maticu. Jedna možnosť je vybrať iba také časti zarovnaní, ktoré sú viac zachované a tým pádom sa zarovnávajú rovnako pre širšiu škálu použitých matíc. Druhá možnosť je použiť zarovnania, v ktorých napríklad dobre navzájom súhlasí trojrozmerná štruktúra zarovnaných proteínov.

Skórovaciu maticu však môžeme odvodiť aj z čisto teoretických modelov H a R, ktoré neboli odpozorované zo žiadnych dát. Uvažujme napríklad model R, v ktorom sú všetky štyri bázy rovnako pravdepodobné, t.j. $p(\mathtt{A}) = p(\mathtt{C}) = p(\mathtt{G}) = p(\mathtt{T}) = 1/4$. V modeli H budeme mať výskyt dvoch rovnakých báz celkovú pravdepodobnosť α, pričom každá z dvojíc `AA`, `CC`, `GG`a `TT` má pravdepodobnosť $\alpha/4$. Podobne každá z 12 rôznych dvojíc `AC`, `AG`, ..., `TG`má pravdepodobnosť $(1-\alpha)/12$. Potom skóre dvojice rovnakých báz (napr. `AA`) bude $\log \frac{\alpha/4}{1/16} = \log 4\alpha$. Podobne skóre pre rôzne bázy (napr. `AC`) bude $\log \frac{(1-\alpha)/12}{1/16} = \log 4(1-\alpha)/3$. Napríklad pre sekvencie, ktoré sú 75% identické ($\alpha = 3/4$) dostávame náš starý známy skórovací systém, ktorý dáva $+1$ za zhodu a -1 za nezhodu.

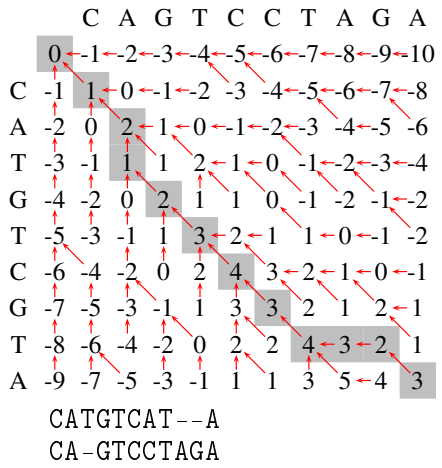
2.3.2 Afínne skórovanie medzier

Doteraz sme sa zaoberali skórovaním stĺpcov bez medzier. V jednoduchom skórovaní sme za každú pomlčku dali rovnakú penaltu -1. Na skórovanie medzier sa však zvykne používať trochu zložitejší spôsob, ktorý berie do úvahy to, že počas evolúcie sa môže naraz zmazať viacero susedných báz. Pri *afínnom skórovaní medzier* máme dve penalty: *skóre za začatie medzery* (*gap opening cost*) o a *skóre za rozšírenie medzery* (*gap extension cost*) e. V zarovnaní na obrázku 2.1 je skupina troch pomlčiek vedľa seba. Cena prvej z týchto troch pomlčiek bude o a cena každej ďalšej e, celková cena medzery dĺžky tri je teda $o+2e$. Vo všeobecnosti má medzera dĺžky g skóre $o+(g-1)e$. Zvyčajne používame o menšie ako e. Napríklad základné nastavenia programu BLAST pre DNA sekvencie sú zhoda $+2$, nezhoda -3, $o = -5$ a $e = -3$. Toto skórovanie zhôd a nezhôd zodpovedá približne modelu H, v ktorom je 88% zarovnaných báz identických.

2.4 Algoritmy na hľadanie zarovnaní

Doteraz sme sa zaoberali tým, ako presne zadefinovať problém lokálneho a globálneho zarovnania a ako skonštruovať systém skórovania. V tejto kapitole si ukážeme algoritmy, ako takto sformulovaný problém naozaj riešiť. Jedna možnosť je prehľadávanie všetkých možností: budeme postupne vytvárať všetky možné zarovnania, pre každé spočítame jeho

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



Obr. 2.4: Príklad matice dynamického programovania pre problém globálneho zarovnania. Vstupné sekvencie sú $X =$ `CATGTCATA` a $Y =$ `CAGTCCTAGA`. Vľavo sú v matici iba hodnoty $A[i, j]$, vpravo aj spätné šípky a cesta zodpovedajúca zarovnaniu s najvyšším skóre. V ľavej matici je zarámované políčko $A[2, 3]$, ktoré zodpovedá zarovnaniu reťazcov `CA` a `CAG`. Najlepšie zarovnanie týchto reťazcov bude obsahovavať dve zhody a jednu pomlčku, takže jeho skóre bude 1, čo je skutočne hodnota uložená v tomto políčku matice.

skóre a zvolíme to zarovnanie, ktoré malo skóre najvyššie. Takýto algoritmus by však bol veľmi pomalý, lebo počet rôznych zarovnaní exponenciálne rastie s dĺžkou zarovnávaných sekvencií. Ukážeme si však, ako tento problém riešiť efektívne technikou dynamického programovania. Pre dlhé sekvencie ani dynamické programovanie nie je dosť rýchle, preto sa v praxi používajú heuristické algoritmy, ktoré si tiež v krátkosti predstavíme.

2.4.1 Dynamické programovanie

Najskôr si ukážeme algoritmus na spočítanie globálneho zarovania, ktorý sa nazýva Needlemanov a Wunschov algoritmus (Needleman and Wunsch, 1970). Pri globálnom zarovnaní sa snažíme zarovnať všetky znaky sekvecnie $X = x_1x_2 \ldots x_n$ a všetky znaky sekvencie $Y = y_1y_2 \ldots y_m$. Budeme používať iba jednoduché skórovanie, ktoré dá zhode $+1$, nezhode -1 a pomlčke tiež -1.

Algoritmus postupne vypĺňa dvojrozmernú tabuľku A, ktorej riadky zodpovedajú jednotlivým bázam sekvencie X a stĺpce jednotlivým bázam sekvencie Y. Na políčku $A[i, j]$ tejto tabuľky bude skóre najlepšieho globálneho zarovnania pre prvých i báz sekvencie X a prvých j báz sekvencie Y (pozri príklad takejto matice na obrázku 2.4). Teda namiesto toho, aby sme problém globálneho zarovnania riešili iba pre jednu dvojicu X a Y, riešime ho pre všetky dvojice, ktoré z X a Y získame uvažovaním iba niekoľkých báz zo začiatku sekvencie. Ak však poznáme takéto hodnoty pre kratšie sekvencie, ľahko z nich skombinujeme hodnotu pre trochu dlhšiu sekvenciu, takže celkové vypĺňanie tabuľky pôjde pomerne rýchlo.

Ako špeciálny okrajový prípad budeme uvažovať aj hodnoty $A[i, j]$, keď i alebo j je nula, teda ak je jedna z dvoch sekvencií prázdna. Ak máme zarovnať sekvenciu $x_1x_2 \ldots x_i$

s prázdnou sekvenciou, potrebujeme pod každú z i báz vložiť pomlčku, za ktorú dostaneme pokutu -1. Celkové skóre teda bude $A[i,0] = -i$. Podobne $A[0,j] = -j$, lebo zarovnávame prvých j báz z Y s pomlčkami.

Touto úvahou teda vieme vyplniť nultý stĺpec a nultý riadok matice A. Ak chceme spočítať nejaké políčko $A[i,j]$ vo vnútri matice, potrebujeme si uvedomiť, ako môže vyzerať posledný stĺpec najlepšieho zarovnania pre $x_1x_2\ldots x_i$ a $y_1y_2\ldots y_j$. Sú iba tri možnosti: buď bude obsahovať x_i aj y_j, alebo x_i zarovanané s pomlčkou, alebo y_j zarovnané s pomlčkou. Ak posledný stĺpec tohto zarovnania obsahuje x_i a y_j, cenu tohto stĺpca vieme spočítať: je to $+1$ ak $x_i = y_j$ alebo -1 ak sa nerovnajú. Ak by sme tento posledný stĺpec zmazali, dostali by sme zarovnanie pre sekvencie $x_1\ldots x_{i-1}$ a $y_1\ldots y_{j-1}$. A nie hocijaké, musí ísť o najlepšie zarovnanie týchto dvoch sekvencií. Ak by sa totiž dalo skóre tohto zarovania vylepšiť, po pridaní stĺpca s x_i a y_j by sme zlepšili aj skóre celého zarovnania, ale predpokladáme, že to je najlepšie možné. Cena najlepšieho zarovnania pre $x_1\ldots x_{i-1}$ a $y_1\ldots y_{j-1}$ je $A[i-1,j-1]$ a teda cena najlepšieho zarovnania pre $x_1x_2\ldots x_i$ a $y_1y_2\ldots y_j$ je $A[i,j] = A[i-1,j-1] + s(x_i,y_j)$, kde pre jednoduchosť ako $s(x,y)$ označíme skóre stĺpca obsahujúceho bázy x a y ($+1$ aleb -1).

Celá táto úvaha ale platí, len ak posledný stĺpec zarovnania obsahuje x_i a y_j. Ak posledný stĺpec obsahuje x_i zarovanané s pomlčkou, skóre tohto stĺpca bude -1 a po jeho zmazaní dostaneme zarovnanie $x_1\ldots x_{i-1}$ a $y_1\ldots y_j$ so skóre $A[i-1,j]$. Skóre celého pôvodného zarovnania je teda $A[i,j] = A[i-1,j] - 1$. Ak nastane posledný prípad, t.j. posledný stĺpec obsahuje y_j zarovnané s medzerou, skóre zarovnania je $A[i,j] = A[i,j-1] - 1$.

Máme teda tri vzorce na výpočet hodnoty $A[i,j]$ z iných hodnôt, ale nevieme, ktorý z nich použiť, lebo nevieme, aký je posledný stĺpec najlepšieho zarovnania. Stačí však zobrať tú z týchto troch hodnôt, ktorá je najvyššia, lebo všetky tri zodpovedajú skutočným zarovnaniam a my hľadáme zarovnanie s maximálnym skóre. Dostávame teda nasledujúci rekurentný vzťah na výpočet $A[i,j]$:

$$A[i,j] = \max \begin{cases} A[i-1,j-1] + s(x_i,y_j) \\ A[i-1,j] - 1 \\ A[i,j-1] - 1 \end{cases}$$

Políčko $A[i,j]$ v matici teda vieme spočítať ako kombináciu hodnôt v políčkach $A[i-1,j]$, ktoré je hneď nad ním, $A[i,j-1]$, ktoré je naľavo od neho a $A[i-1,j-1]$, ktoré je šikmo vľavo hore. Celú maticu môžeme teda vypĺňať po riadkoch zhora nadol a každý riadok zľava doprava (algoritmus 1). Keď máme maticu A celú vyplnenú, pravé dolné políčko $A[n,m]$ obsahuje skóre najlepšieho zarovnania sekvencií X a Y. Nás však často zaujíma nielen skóre, ale aj samotné zarovnanie. Aby sme ho spočítali, potrebujeme si počas vypĺňania matice A pre každé políčko $A[i,j]$ pamätať, ktorá z hodnôt $A[i-1,j-1]$, $A[i-1,j]$ a $A[i,j-1]$ dosiahla maximálnu hodnotu. Na obrázku 2.4 sme to vyznačili spätnými šípkami, v algoritme 1 si ukladáme tieto šípky do matice B. Takáto spätná šípka určuje, aký má byť posledný stĺpec zarovnania: ak ide šikmo hore, posledný stĺpec obsahuje dve bázy, ak ide hore, posledný stĺpec obsahuje bázu zo sekvencie X zarovnanú s pomlčkou a ak ide doľava, posledný stĺpec obsahuje bázu z Y zarovnanú s pomlčkou. Môžeme teda prejsť po šípkach z pravého dolného do ľavého horného rohu matice a stĺpec po stĺpci od konca vypisovať zarovnanie.

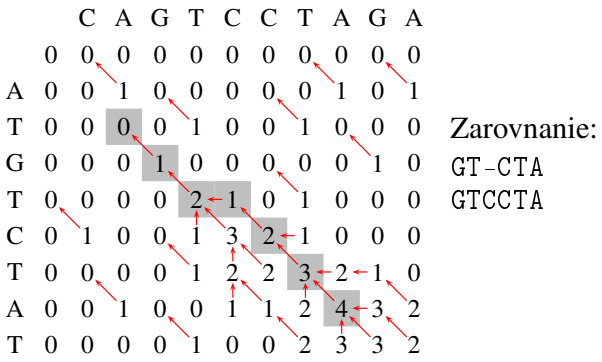
Časová zložitosť algoritmu 1 je $O(nm)$, lebo musíme vyplniť nm políčok a každé

```
// Vyplnenie nultého riadku a stĺpca matíc A a B
foreach i ∈ {0, ..., n} do
    A[0, i] ← −i; A[i, 0] ← −i;
    B[0, i] ← vľavo; B[i, 0] ← hore;
end
// Vyplnenie matíc A a B
foreach i ∈ {1, ..., n} do
    foreach j ∈ {1, ..., m} do
        A[i, j] ← max {A[i − 1, j − 1] + s(x_i, y_j), A[i − 1, j] − 1, A[i, j − 1] − 1};
        if A[i, j] = A[i − 1, j − 1] + s(x_i, y_j) then B[i, j] ← šikmo;
        if A[i, j] = A[i − 1, j] − 1 then B[i, j] ← hore;
        if A[i, j] = A[i, j − 1] − 1 then B[i, j] ← doľava;
    end
end
// Vypisovanie zarovnania, spätný chod po šípkach
i ← n; j ← m;
while i > 0 or j > 0 do
    if B[i, j] = šikmo then
        pridaj stĺpec x_i, y_j na začiatok zarovnania;
        i ← i − 1; j ← j − 1;
    end
    if B[i, j] = hore then
        pridaj stĺpec x_i, − na začiatok zarovnania;
        i ← i − 1;
    end
    if B[i, j] = doľava then
        pridaj stĺpec −, y_j na začiatok zarovnania;
        j ← j − 1;
    end
end
```

Algorithm 1: Needlemanov a Wunschov algoritmus.

spočítame v konštantom čase. V základnej verzii algoritmu potrebujeme tiež ukladať celú maticu veľkosti $(n+1) \times (m+1)$, čo môže byť pre väčšie sekvencie problém. Existuje však aj modifikácia tohto algoritmu, ktorá má zhruba dvojnásobný čas výpočtu, ale vyžaduje iba pamäť $O(m+n)$ (Hirschberg, 1975).

Doteraz sme riešili problém globálneho zarovnania, kde sa snažíme zarovnať vstupné sekvencie v celej ich dĺžke. Pri lokálnom zarovnaní hľadáme vo vstupných sekvenciách len nejaké podobné úseky, ktoré môžu začať aj skončiť na ľubovoľnom mieste v sekvencii. Problém lokálneho zarovania môžeme riešiť Smithovým a Watermanovým algoritmom Smith and Waterman (1981), ktorý sa od Needlman-Wunschovho algoritmu líši len v niekoľkých malých detailoch. Opäť budeme vypĺňať maticu A, ktorej prvok $A[i, j]$ bude najvyššie skóre lokálneho zarovnania medzi reťazcami $x_1, \ldots, x_i$ a $y_1, \ldots, y_j$, ktoré buď obsahuje bázy x_i aj y_j (nie nutne zarovnané spolu), alebo je prázdne (obr. 2.5). Vzťah na výpočet $A[i, j]$ sa od algoritmu pre globálne zarovnania líši iba v tom, že na ľubovoľnom mieste uvažujeme aj prázdne zarovnanie so skóre 0 (v matici teda nebudú žiadne záporné



Obr. 2.5: Príklad matice dynamického programovania pre problém lokálneho zarovnania. Vstupné sekvencie sú $X =$ `ATGTCTAT` a $Y =$ `CAGTCCTAGA`.

čísla):

$$A[i,j] = \max \begin{cases} 0 \\ A[i-1,j-1] + s(x_i, y_j) \\ A[i-1,j] - 1 \\ A[i,j-1] - 1 \end{cases}$$

Po vyplnení celej matice v nej nájdeme najväčšiu hodnotu. Tá predstavuje skóre najlepšieho lokálneho zarovnania. Zarovnanie tiež vypisujeme pomocou spätných šípok uložených v matici B, ale začneme vypisovať z políčka, v ktorom sme našli najväčšiu hodnotu a skončíme, keď prídeme na políčko s hodnotou nula (ktoré predstavuje začiatok zarovnania). Časová zložitosť je rovnako ako pri globálnom zarovnaní $O(mn)$.

Obidva algoritmy sme popísali pre jednoduché skórovanie, môžeme ich však použiť aj pre zložitejšie skórovacie matice, ako napríklad BLUSUM62. Stačí vo výraze $s(x, y)$ použiť príslušnú hodnotu z matice pre nukleotidy alebo amino kyseliny x a y. Afínne skórovanie medzier vyžaduje trochu väčšiu zmenu algoritmu, tiež však funguje v čase $O(mn)$.

Programy na lokálne zarovnávanie často nájdu nielen jedno zarovnanie s najvyšším skóre, ale viacero zarovnaní medzi rôznymi časťami sekvencie. Preto po vypísaní prvého lokálneho zarovnania s najväčším skóre môžeme v iných častiach matice hľadať ďalšie políčka s vysokým skóre a prechodom po spätných šípkach vypisovať ďalšie zarovnania.

2.4.2 Heuristické algoritmy na hľadanie zarovnaní

Aj keď algoritmy dynamického programovania pre lokálne a globálne zarovnanie sekvencií pracujú v polynomiálnom čase, v praxi pre dlhšie sekvencie nie sú dostatočne rýchle. Ako vidíme v tabuľke 2.1, na porovnanie dvoch sekvencií dĺžky 100 miliónov báz by sme na jednom počítači potrebovali 25 rokov. Pritom to je približne dĺžka jedného ľudského chromozómu a často chceme porovnávať oveľa väčšie databázy sekvencií z viacerých organizmov, takže časové nároky by boli ešte oveľa väčšie. Čas výpočtu môžeme o niečo znížiť použitím rýchlejšieho počítača, rýchlejšou implementáciou, alebo rozdelením práce medzi viacero počítačov. Nakoľko však čas rastie kvadraticky s dĺžkou sekvencie, zrýchlením výpočtu štvornásobne môžeme za určitý čas spracovať len dvojnásobne dlhé sekvencie. V praxi sa namiesto dynamického programovania používajú heuristické algoritmy, ktoré nezaručujú, že nájdu najlepšie možné zarovnanie, ale bežia oveľa rýchlejšie, takže je re-

Tabuľka 2.1: Čas behu Smithovho a Watermanovho algoritmu na bežnom kancelárskom počítači z roku 2006. Algoritmus bol implementovaný v jazyku C++ priamočiarym spôsobom a spúšťaný na dvoch rovnako dlhých náhodne vygenerovaných DNA sekvenciách. Hodnoty označené hviezdičkou boli extrapolované z hodnôt pre menšie hodnoty n.

Dĺžka sekvencie	Čas výpočtu
100	0.0008s
1,000	0.08s
10,000	8s
100,000	13 minút (*)
1,000,000	22 hodín (*)
10,000,000	3 mesiace (*)
100,000,000	25 rokov (*)

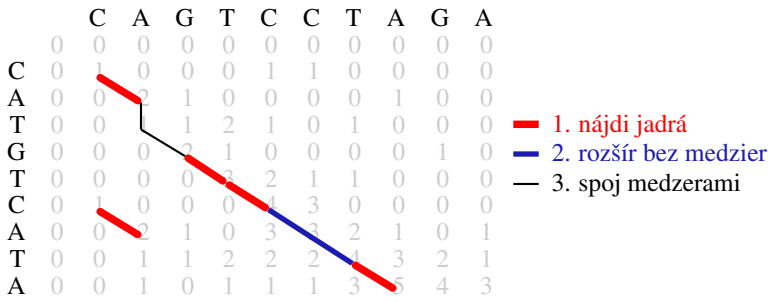
alistické ich používať aj na porovnávanie celých genómov, či ešte väčších databáz. Ich základnou ideou je na základe predspracovania vstupných sekvencií vytipovať sľubné časti matice dynamického programovania a vyplnia sa iba hodnoty v týchto častiach, zvyšok matice sa ignoruje.

Asi najznámejším príkladom algoritmu tohto typu je nukleotidový BLAST (Altschul et al., 1990), ktorý hľadá lokálne zarovnania medzi dvomi DNA sekvenciami. Tento program má dôležitý parameter *veľkosť slova* (*word size*) W. Typické nastavenie je napríklad $W = 11$, ale používateľ ho môže zmeniť. Algoritmus postupuje v najsledujúcich krokoch (ilustrácia na obrázku 2.6).

- **Krok 1:** Nájde všetky úseky dĺžky W, ktoré sa zhodujú medzi X a Y. Tieto úseky budeme nazývať *jadrá zarovnania*.
- **Krok 2:** Každé jadro sa pokúsi rozšíriť pozdĺž diagonály do dlhšieho zarovnania bez medzier.
- **Krok 3:** Zarovnania bez medier, ktoré sú na neďalekých uhlopriečkach spojí pomocou medzier do dlhšieho zarovnania.
- **Krok 4:** Vyplní dynamickým programovaním oblasť okolo každého nájdeného zarovnania s cieľom mierne vylepšiť skóre (tento krok je prípadne možné vynechať).
- **Krok 5:** Vypíše zarovnania s dostatočne vysokým skóre.

Aby bol algoritmus dostatočne rýchly, musíme vedieť rýchlo nájsť jadrá zarovnaní. Algoritmus najskôr vezme prvú sekvenciu X a zostaví slovník všetkých *slov*, t.j. úsekov dĺžky W, ktoré sa v nej nachádzajú. Pre každé slovo si pamätáme všetky pozície, kde sa v X nachádza (obr. 2.7). Potom prechádzame cez všetky slová dĺžky W v Y a hľadáme každé vo vytvorenom slovníku. Ak ho tam nájdeme, vieme aj kde v X sa nachádza a to nám dáva jadrá zarovnaní: úseky (slová) dĺžky W, ktoré sa vyskytujú v X aj Y. Na rýchle vytváranie a vyhľadávanie v slovníku existujú v informatike mnohé štandardné dátové štruktúry.

Keďže jadrá zarovnaní vieme nájsť pomerne rýchlo, najviac nás spomalia ďalšie kroky algoritmu, ktoré sa pokúšajú rozšíriť nájdené jadrá do dlhších zarovnaní. Ak však zvolíme



Obr. 2.6: Ilustrácia algoritmu z programu BLAST pre $W = 2$ (v praxi sa používajú vyššie hodnoty W). V pozadí je celá matica Smithovho a Watermanovho algoritmu, rôznymi farbami sú naznačené časti zarovnania nájdené v rôznych fázach algoritmu. Algoritmus však nevytvára v pamäti celú maticu, pracuje len s nájdenými jadrami a zarovnaniami.

Slovník pre sekvenciu $X =$ `CAGTCCTAGA`:

Slovo	`AG`	`CA`	`CC`	`CT`	`GA`	`GT`	`TA`	`TC`
Výskyty v X	2, 8	1	5	6	9	3	7	4

Hľadanie slov zo sekvencie $Y =$ `CATGTCATA` v slovníku:

Slovo z Y	1:`CA`	2:`AT`	3:`TG`	4:`GT`	5:`TC`	6:`CA`	7:`AT`	8:`TA`
Nájdené výskyty v X	1	–	–	3	4	1	–	7

Obr. 2.7: Príklad hľadania jadier zarovnaní v programe BLAST pre $W = 2$. Napríklad slovo GT na pozícii 3 v X a na pozícii 4 v Y tvorí jedno jadro.

malé W, nájdeme veľa jadier, ktoré sú čisto náhodnou zhodou a nepatria do žiadneho zarovnania s vyšším skóre. Každé takéto jadro sa program pokúsi rozšíriť na väčšie zarovnanie, ale nakoľko nenájde dosť vysoké skóre, nič nevypíše. Takéto náhodne sa vyskytujúce jadrá nemajú teda vplyv na kvalitu výstupu, ale spomaľujú program. Intuitívne, čím menšiu hodnotu W použijeme, tým viac budeme mať náhodne sa vyskytujúcich jadier mimo skutočné zarovnania a tým bude program pomalší.

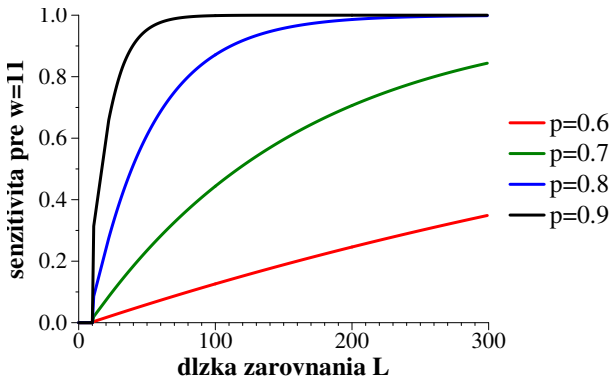
Naopak pri väčších hodnotách W sa nám nepodarí nájsť zarovnania, ktoré nikde neobsahujú W zhodných báz vedľa seba. Ak napríklad uvažujeme zarovnanie dĺžky 100, v ktorom sa 80% báz zhoduje, ale nezhodujúce sa bázy su rozmiestnené pomerne rovnomerne, očakávame zhruba jednu nezhodu každých päť báz. Môže sa teda ľahko stať, že nikde v tomto zarovnaní nebude 11 zhodujúcich sa báz za sebou a teda toto zarovnanie s nastavením $W = 11$ nenájdeme. Pojmom *senzitivita* algoritmu rozumieme aké percento skutočných zarovnaní algoritmus nájde. Intuitívne sme teda dospeli k záveru, že so znižujúcim W bude BLAST senzitívnejší, ale pomalší a naopak so zvyšujúcim sa W bude rýchlejší, ale nájde menej zarovnaní.

Na presnejšie odhady vplyvu zmeny veľkosti slova na rýchlosť a senzitivitu môžeme opäť použiť jednoduché pravdepodobnostné modely. Ako sme už písali vyššie, rýchlosť súvisí s počtom náhodne sa vyskytujúcich jadier medzi nesúvisiacimi časťami sekvencie. Aby sme počet takýchto náhodných jadier odhadli, budeme predpokladať, že zarovnávame náhodne generované DNA sekvencie dĺžok m a n. Pravdepodobnosť, že sa dve bázy z týchto sekvencií budú rovnať, je $1/4$. Aby sme mali jadro, potrebujeme W takýchto zhodných báz za sebou a teda pravdepodobnosť, že na danej pozícii i v X a pozícii j v Y začína jadro je $(1/4)^W$. Celkovo máme približne nm dvojíc pozícií, kde jadro môže začať a každá z nich má rovnakú pravdepodobnosť. Priemerný počet výskytov jadra v takýchto náhodných sekvenciách je teda približne $mn/4^W$. (V skutočnosti je ich o trochu menej, lebo sme zanedbali fakt, že tesne pri konci sekvencie sa už výskyt jadra nezmestí, takže počet výskytob jadra je iba $(m - W + 1)(n - W + 1)/4^W$.) Zvýšením W o jedna teda priemerný počet náhodných jadier klesne zhruba štvornásobne.

Na odhad senzivity použijeme model zarovnania, ktoré má nejakú pevnú dĺžku L a každý stĺepc tohto zarovnania je zhoda s pravdepodobnosťou p (napr. $p = 0.7$) a nezhoda s pravdepodobnosťou $1 - p$ (medzery pre jednoduchosť neuvažujeme). Takéto zarovnanie môže byť nájdené programom BLAST, len ak obsahuje W zhôd za sebou a pravdepodobnosť, že sa tak stane v náhodnom zarovnaní z tohto modelu bude teda náš odhad senzitivity. Výpočet tejto pravdepodobnosti je o niečo komplikovanejší ako v prípade počtu náhodných jadier. Neuvádzame preto vzorec, ale na obr. 2.8 sú vykreslené hodnoty pre rôzne dĺžky zarovnania a pre rôzne hodnoty p. Ako vidíme, pri zarovnaniach so 60% a 70% zhôd aj dlhé zarovnania majú pomerne malú šancu, že ich BLAST objaví.

V týchto jednoduchých odhadoch senzitivity a počtu náhodných jadier sme zanedbali rôzne javy, ktoré sa v biologických sekvenciách exitujú. Môžeme napríklad brať do úvahy nerovnomerné frekvencie výskytu nukleotidov, závislosti medzi susednými bázami v sekvencii, či oblasti zarovnaní s vyššou a nižšou mierou mutácií. Tento posledný faktor často umožní BLASTu nájsť viac zarovnaní, než by sme očakávali v našom jednoduchom modeli, lebo homologické sekvencie často obsahujú silne zachované funkčné miesta s veľa zhodami pri sebe.

Hoci techniky popísané vyššie sa dajú použiť aj na zarovnávanie proteínov, často chceme zarovnávať proteíny s pomerne malým podielom zhodných amino kyselín, kde by



Obr. 2.8: Senzitivita BLASTu s nastavením $W = 11$ pri náhodných zarovnaniach s pravdepodobnosťou zhody p a dĺžkou L.

```
>ref|YP_003983317.1|
hypothetical protein HMPREF0733_10425 [Rothia dentocariosa ATCC 17931]
Length=414

 Score = 30.3 bits (64),  Expect = 2.6
 Identities = 8/11 (73%), Positives = 8/11 (73%), Gaps = 0/11 (0%)

Query  3    ITHWATERMAN  13
            IT W TERM N
Sbjct  92   ITYWTTERMEN  102
```

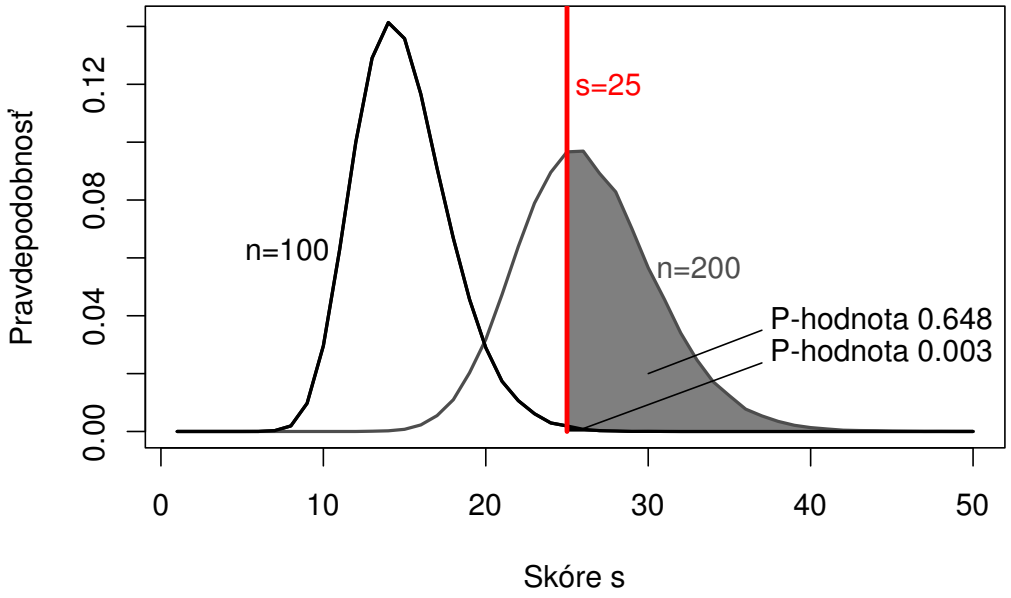
Obr. 2.9: Zarovnanie s najvyšším skóre medzi výsledkami vyhľadávania reťazca `SMITHWATERMAN` v neredundantnej databáze proteínov na `http://blast.ncbi.nlm.nih.gov/`. Aj keď náš reťazec zjavne nie je biologického pôvodu, BLAST našiel zarovnanie s 73% zhôd. E-hodnota nájdeného zarovnania je však až 2,6, takže ho nemôžeme považovať za štatisticky významné.

sme dosiahli iba veľmi malú sekzitivitu. Proteínový BLAST na rozdiel od nukleotidového BLASTu nevyžaduje ako jadro zarovnania presnú zhodu dĺžky W, ale vyžaduje v úseku dĺžky W dosiahnutie určitého skóre. Napríklad môže vyžadovať v úseku 3 aminokyselín skóre aspoň 13 napríklad v matici BLOSUM62. Ak sekvencia X obsahuje trojicu NIL a sekvencia Y obsahuje trojicu NLR, ich BLOSUM65 skóre 6+2+5=13 a teda by tvorili jadro pri použití tohto pravidla. Naopak trojica AIL netvorí jadro ani v kombinácii sama so sebou, lebo každá z týchto aminokyselín má pri zhode skóre 4 a teda celkové skóre je iba 12.

2.5 Štatistická významnosť zarovnaní

Základný Smithov a Watermanov algoritmus nájde najlepšie lokálne zarovnanie pre ľubovoľné dve sekvencie, ktoré mu dáme (obr. 2.9). Užívateľ sa však musí rozhodnúť, či nájdené zarovnanie vyzerá dostatočne vierohodne na to, aby predstavovalo skutočnú podobnosť sekvencií. Ako vodídko v tomto rozhodnutí sa používajú indikátory *štatistickej významnosti* zarovnania: *P-hodnota* (*P-value*) alebo *E-hodnota* (*E-value*).

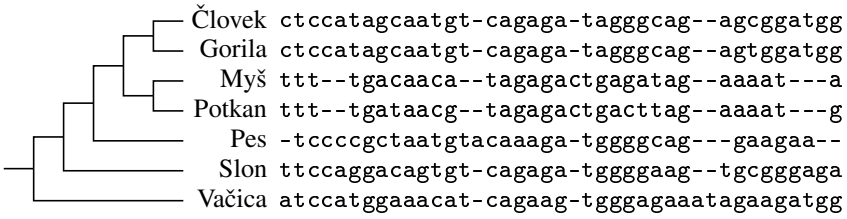
Uvažujme, že sme našli zarovnanie so skóre s medzi sekvenciami (alebo databázami



Obr. 2.10: P-hodnota lokálneho zarovnania so skóre $s = 25$ medzi dvoma sekvenciami dĺžky $n = 100$ alebo $n = 200$ (skórovanie $+1$ zhoda, -1 nezhoda alebo medzera). Rozdelenie hodnôt skóre bolo získané zarovnávaním 100 000 párov náhodných sekvencií. Pri $n = 100$ je P-hodnota približne 0.003 a zodpovedá malej čiernej ploche pod krivkou napravo od zvislej čiary pre $s = 25$. Pri dlhších sekvenciách zodpovedá P-hodnota 0.648 veľkej sivej ploche napravo od zvislej čiary. Pri takto dlhých sekvenciách teda očakávame skóre 25 a vyššie vo viac ako 60% prípadov čisto náhodou a teda nejde o štatisticky významné zarovnanie.

sekvencií) dĺžky n a m. P-hodnota tohto zarovnania je pravdepodobnosť, že medzi náhodne generovanými sekvenciami tej istej dĺžky by sme našli tiež zarovnanie so skóre s alebo vyšším. Predstavme si teda, že opakujeme nasledujúci experiment: vygenerujeme dve náhodné sekvencie dĺžky n a m, nájdeme najlepšie lokálne zarovnanie medzi nimi a poznačíme si jeho skóre. Po veľa opakovaniach dostaneme pravdepodobnostné rozdelenie skóre a pýtame sa, kde v tomto rozdelení leží hodnota s, ktorá reprezentuje skóre nájdeného zarovnania medzi skutočnými sekvenciami (obr. 2.10). P-hodnota je plocha pod krivkou od s vyššie a hľadáme zarovnania, kde je táto hodnota malé číslo (blízke k nule). Ak je totiž hodnota s niekde v pravom chvoste zarovnania, tak len veľmi zriedkavo dostaneme v našom náhodnom experimente také vysoké skóre a naše pôvodne nájdené zarovnanie môžeme považovať za štatisticky významné. Ak naopak pri experimente často dostávame rovnako skóre s alebo ešte vyššie hodnoty, nájdené zarovnanie so skóre s medzi skutočnými sekvenciami môže byť len výsledkom náhodnej podobnosti bez biologického významu.

Treba si uvedomiť, že P-hodnota zarovnania závisí od dĺžok sekvencií: čím dlhšie sekvencie, tým väčšia pravdepodobnosť, že sa v nich niečo nájde čisto náhodou. Ak vyhľadávame jednu sekvenciu vo väčšej databáze, uvažujeme ako m dĺžku celej databázy,



Obr. 2.11: Viacnásobné zarovnanie DNA sekvencií z niekoľkých cicavcov. Prvá medzera v sekvencii človeka pravdepodobne zodpovedá inzercii bázy A v sekvencii psa, lebo táto báza sa v iných zobrazených cicavcoch nevyskytuje. Naopak prvá medzera v sekvencii myši a potkana zrejme zodpovedá delécii v ich spoločnom predkovi. Posledná medzera v sekvencii človeka môže rovnako dobre zodpovedať inzercii u vačice alebo delécii u spoločného predka ostatných zobrazených cicavcov.

nielen tej jednej sekvencie, v ktorej sa nakoniec našlo zarovnanie.

Nakoľko P-hodnota závisí od skóre s, ako aj od dĺžok oboch sekvencií, potrebovali by sme ju v praxi počítať po každom zarovnávaní dvoch nových sekvencií. Bolo by však veľmi časovo náročné robiť to generovaním veľkého množstva náhodných sekvencií a hľadaním zarovnaní medzi nimi. Namiesto toho sa používajú matematicky odvodené vzorce na odhad tejto hodnoty (Karlin and Altschul, 1990; Mitrophanov and Borodovsky, 2006).

Namiesto P-hodnoty sa pri hľadaní v databáze často používa E-hodnota, ktorá vyjadruje priemerný počet zarovnaní so skóre aspoň s alebo vyšším medzi náhodne generovanými sekvenciami dĺžok n a m. Teda pri P-hodnote nás zaujíma pravdepodobnosť, že náhodne vznikne aspoň jedno zarovnanie so skóre aspoň s, ale už nás nezaujíma, koľko ich bude. Pri E-hodnote rátame priemerný počet takých zarovnaní, takže na rozdiel od P-hodnoty, E-hodnota môže byť aj viac ako jedna. Ak je E-hodnota väčšia ako jedna, čisto náhodou by sme očakávali aspoň jedno také silné zarovnanie a teda zarovnania s tak vysokou E-hodnotou rozhodne nebudeme považovať za štatisticky významné. V štatistike sa v rôznych testoch štadardne používajú prahy na P-hodnotu 0.05 alebo 0.01, ale pri zarovnávaní sekvencií často používame ešte opatrnejší prah a uvažujeme len zarovnania s P-hodnotou menšou ako napríklad 10^{-5}. Pri malých hodnotách blízkych nule sú E-hodnota a P-hodnota približne rovnako veľké, teda taký istý prah môžeme aplikovať aj na E-hodnotu.

2.6 Viacnásobné zarovnania

Doteraz sme vždy navzájom zarovnávali dve sekvencie, v bioinformatike sa však často používajú aj viacnásobné zarovnania. Viacnásobné zarovnania sa používajú na rekonštrukciu fylogenetických stromov. Vieme z nich tiež v niektorých prípadoch usudzovať, či medzery v zarovnaní zodpovedajú inzerciám alebo deléciám (obr. 2.11). Pomáhajú nám tiež určiť, ktoré časti sekvencie sa menia pomaly a teda možno majú nejakú dôležitú funkciou v bunke.

Pri viacnásobnom zarovnaní väčšinou pracujeme so sekvenciami, o ktorých sme už vopred určili, že sú pravdepodobne homologické a snažíme sa ich zarovnať v ich celej dĺžke, t.j. globálne. Podobne ako pri zarovnávaní dvoch sekvencií, aj teraz si musíme najskôr zvoliť nejaký konkrétny systém skórovania zarovnaní. Pri skórovaní dvoch sekvencií sme

Viacnásobné zarovnanie:

```
Sekvencia 1:  A  C  A  G  T  -  A
Sekvencia 2:  A  -  A  C  T  -  A
Sekvencia 3:  G  C  A  C  T  G  A
```

Párové zarovnania:

```
Sekvencia 1:  A  C  A  G  T  A
Sekvencia 2:  A  -  A  C  T  A     Skóre 2
-------------------------------------------
Sekvencia 1:  A  C  A  G  T  -  A
Sekvencia 3:  G  C  A  C  T  G  A  Skóre 1
-------------------------------------------
Sekvencia 2:  A  -  A  C  T  -  A
Sekvencia 3:  G  C  A  C  T  G  A  Skóre 1
```

Obr. 2.12: Príklad skórovania viacnásobného zarovnania metódou sčítavania skóre párov sekvencií. Ak pre párové zarovnania skórujeme zhodu $+1$, nezhodu -1 a pomlčku -1, dostávame skóre párových zarovnaní 2, 1, 1 a teda celkové skóre viacnásobného zarovnania bude 4.

použili jednoduché pravdepodobnostné modely popisujúce frekvencie rôznych mutácií. Pri viacerých sekvenciách by sme ideálne tiež chceli použiť model evolúcie, ktorý by bral do úvahy evolučné vzťahy medzi sekvenciami, pravdepodobnosti výskytu rôznych mutácií a podobne. O takýchto modeloch sa viac dozvieme v kapitole 3. Na to, aby sme zarovnanie skórovali pomocou evolučného modelu by sme ale potrebovali vedieť fylogenetický strom zarovnávaných sekvencií. Stromy sa však zostavujú práve na základe viacnásobných zarovnaní, čím sa dostávame do začarovaného kruhu. Existujú metódy, ktoré odhadujú strom aj zarovnanie súčasne, aby čo najlepšie sedeli s evolučným modelom. Takéto metódy však bývajú väčšinou pomerne pomalé.

Používajú sa však aj metódy, ktoré priamočiaro rozširujú skórovanie z dvoch sekvencií na viacero. Ak vezmeme dva riadky viacnásobného zarovnania, dostaneme zarovnanie dvoch sekvencií. Jediným problémom sú stĺpce, v ktorých majú obe sekvencie pomlčky a také stĺpce budeme jednoducho ignorovať. Dostaneme teda zarovnanie dvoch sekvencií, ktorému môžeme priradiť skóre ľubovoľným systémom popísaným vyššie. Celkové skóre viacnásobného zarovnania dostaneme sčítaním skóre zarovnaní pre všetky dvojice sekvencií (*sum of pairs*), ako vidíme v príklade na obrázku 2.12.

Dynamické programovanie na globálne zarovnávanie dvoch sekvencií je možné rozšíriť aj na zarovnávanie viacerých sekvencií (pri sum of pairs skórovaní), problém však je, že čas aj potrebná pamäť rastú exponenciálne s počtom sekvencií. Napríklad pre tri sekvencie dynamické programovanie používa trojrozmernú tabuľku, ktorej prvok $A[i, j, k]$ obsahuje skóre najlepšieho zarovnania pre prvých i báz z prvej sekvencie, prvých j báz z druhej sekvencie a prvých k báz z tretej sekvencie. Navyše pri počítaní hodnoty tohto políčka musíme uvažovať sedem možností, kde môžu byť v poslednom stĺpci zarovnania umiestnené pomlčky (namiesto troch možností, ktoré sme mali pre dve sekvencie). Pre všeobecný prípad, keď zarovnávame k sekvencií, každá dĺžky n, bude časová zložitosť $O(2^k n^k)$. Takýto algoritmus môžeme použiť iba pre veľmý malý počet pomerne krátkych sekvencií.

To, že dynamické programovanie nevieme efektívne zovšeobecniť z dvoch sekvencií na väčší počet neznamená, že to nemôže ísť nejakým iným postupom. O probléme viacnásobného zarovnania však, žiaľ, bolo dokázané, že je NP-ťažký, a teda nie je pravdepodobné,

že ho budeme vedieť rýchlo a presne riešiť (Elias, 2006). V praxi sa používajú heuristiky, ktoré nezaručujú, že nájdu zarovnanie s najvyšším skóre pri zvolenom systéme skórovania, vedia však v rozumnom čase zarovnať aj pomerne veľké množstvo sekvencií.

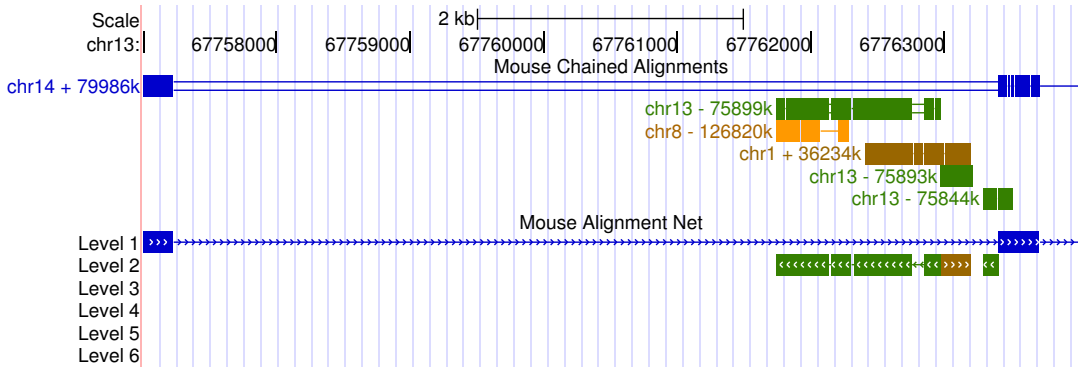
Jednou z takých heuristík je *progresívne zarovnávanie* (*progressive alignment*), kde sa zarovnanie buduje postupne od menších skupín sekvencií k väčším. Väčšinou sa tak deje podľa fylogenetického stromu zarovnávaných sekvencií. Zase tu narážame na začarovaný kruh medzi stromami a zarovnaním, môžeme ho však vyriešiť tak, že použijeme iba približný strom zostavený jednoduchými metódami. Keď máme strom, postupujeme od listov ku koreňu. V našom príklade na obrázku 2.11 by sme najprv zarovnali sekvencie človeka a gorily, potom sekvencie myši a potkana a potom by sme tieto dve zarovnania zarovnali spolu do zarovnania štyroch sekvencií. Potom by sme k tomuto zarovnaniu pridali sekvenciu psa, potom slona a nakoniec vačice. Vždy keď spájame dve zarovnania spolu, nemeníme už umiestnenie medzier vnútri existujúcich zarovnaní, ale iba vzájomné umiestnenie ich stĺpcov.

2.7 Celogenómové zarovnania

Zarovnania sekvencií sú vhodné na úseky genómu, ktoré sa zmenili hlavne lokálnymi mutáciami: substitúciami jednej bázy za inú a kratšími deléciami či inzerciami. V procese evolúcie však dochádza aj k väčším zmenám, keď sa dlhý úsek sekvencie môže premiestniť alebo skopírovať na iné miesto alebo úplne zmazať z genómu. *Celogenómové zarovnanie* (*whole-genome alignment*) je akousi mapou, ktorá pre každý úsek jedného genómu obsahuje zodpovedajúci úsek iného genómu (ak sa ho podarilo nájsť). Často takéto zarovnania nie sú symetrické: jeden genóm zvolíme ako *referenčný genóm* (napríklad genóm človeka) a k nemu zarovnávame jeden alebo viacero genómov *informantov* (napríklad genómy gorily, myši a podobne).

Prvým krokom pri tvorbe celogenómového zarovnania je nájdenie všetkých lokálnych zarovnaní s dostatočne vysokým skóre medzi dvoma genómami. Môže sa stať, že niektoré oblasti referenčného genómu nebudú zarovnané vôbec s ničím a naopak iné oblasti budú zarovnané s viacerými úsekmi genómu informanta. Prvý jav môže nastať preto, že genóm informanta skutočne neobsahuje zodpovedajúci úsek, alebo len v ňom nastalo toľko mutácií, že podobnosť už nie je ľahko rozpoznateľná. Druhý prípad nastáva vtedy, keď došlo v minulosti k duplikácii (skopírovaniu) tohto úseku na iné miesto v genóme. V takom prípade chceme určiť, ktorá z kópií v genóme informanta najlepšie zodpovedá príslušnému úseku referenčného genómu, t.j. že pochádza z tej istej sekvencie v genóme najbližšieho spoločného predka. Takéto úseky nazývame *ortológy* (viac v kapitole 3).

Dôležitým vodítkom pri zostavovaní celogenómových zarovnaní z lokálnych zarovnaní dvoch genómov je *synténia*. Je to zachovanie poradia génov alebo iných úsekov DNA medzi dvoma genómami. Často vidíme skupiny lokálnych zarovnaní, ktoré idú v tom istom poradí v oboch genómoch, ale medzi nimi môžu byť úseky, ktoré buď nie sú zarovnané s ničím, alebo sú zarovnané s inou časťou genómu. V celogenómových zarovnaniach z UCSC prehliadača genómov sa nazývajú takéto skupiny zarovnaní s rovnakým poradím *reťaze* (*chains*) (Kent et al., 2003). Zarovanie, ktoré je časťou dlhšej reťaze, má väčšiu šancu zodpovedať skutočným ortológom, lebo je malá pravdepodobnosť, že to isté poradie génov by v dvoch genómoch vzniklo nezávisle od seba. K jednej časti referenčného genómu



Obr. 2.13: Príklad reťazí a sietí medzi genómom človeka a myši z UCSC prehliadača genómov (Kent et al., 2002) na úseku z ľudského chromozómu 13. Najvrchnejšej úrovni je modrá reťaz zodpovedajúca chromozómu 14 z myši. V medzere tohto zarovnania je niekoľko prekrývajúcich sa reťazí, z ktorých sa do druhej úrovne použila celá zelená reťaz a časti z hnedej a ďalšej zelenej reťaze.

stále ešte môže existovať niekoľko prekrývajúcich sa reťazí. Preto UCSC prehliadač okrem raťazí obsahuje aj *siete* (*nets*), ktoré obsahujú reťaze s čo najvyšším skóre skrátené podľa potreby, aby sa navzájom neprekrývali a každá báza referenčného genómu bola pokrytá najviac jedným zarovnaním. Sieť tvorí hierarchickú štruktúru, kde väčšia medzera v reťazi na najvrchnejšej úrovni môže byť vyplnená reťazou alebo časťou reťaze na nižšej úrovni a medzery v tejto reťazi ďalšími reťazami. Reťaze na nižších úrovniach zodpovedajú častiam genómu, ktoré boli počas evolúcie premiestnené.

2.8 Príklady programov

Na zarovnávanie sekvencií existuje veľké množstvo programov. Mnohé sú prístupné ako služby na internete, ale aj na stiahnutie a na inštaláciu na vlastnom počítači. Medzi najpopulárnejšie programy patrí NCBI BLAST (Altschul et al., 1990, 1997), ktorý umožňuje hľadať lokálne zarovnania medzi užívateľovou sekvenciou a veľkými databázami verejne dostupných sekvencií. Je možné zarovnávať DNA sekvencie (program blastn) alebo proteíny (program blastp). Je však možné zarovnať napríklad aj proteín k DNA sekvenciám (program tblastn), pričom DNA sa preloží do sekvencie aminokyselín podľa genetického kódu a na výsledok sa použije program podobný blastp. Ako sme videli vyššie, BLAST využíva heuristické techniky na zrýchlenie vyhľadávania, takže nemusí vždy nájsť všetky zarovnania s vysokým skóre. Okrem samotných zarovnaní a ich skóre zobrazuje aj štatistickú významnosť (E-hodnotu).

Zaujímavou obmenou štandardného proteínového BLASTu je program PSI-BLAST (Altschul et al., 1997), ktorý často dokáže nájsť aj vzdialenejšie homológy než program BLAST. Postupuje tak, že k zadanej sekvencii najprv nájde blízke homológy štandardným programom blastp. Porovnávaním týchto homológov so zadanou sekvenciou zistí, ktoré pozície sekvencie mutujú viac a ktoré menej. Potom zmení svoj systém skórovania tak, aby nezhoda na viac evolučne zachovanej sekvencii mala horšie skóre ako nezhoda na pozícii, ktorá sa medzi nájdenými homológmi mení častejšie. Tým sa pri vyhľadávaní kladie väčší dôraz na najdôležitejšie časti sekvencie a často je možné týmto spôsobom

nájsť homologické sekvencie s menšou podobnosťou.

Súčasťou UCSC prehliadača genómov je program BLAT (Kent, 2002), ktorý je určený predovšetkým na hľadanie lokálnych zarovnaní medzi veľmi podobnými sekvenciami. BLAT je založený na heuristických technikách, ktoré môžu vynechať mnohé zarovnania medzi menej podobnými sekvenciami, je však veľmi rýchly. Používa sa napríklad na mapovanie osekvenovaných mRNA sekvencií ku genómu alebo porovnávanie veľmi blízko príbuzných genómov. Pri zarovnávaní mRNA sekvencií sa BLAT snaží nájsť správne umiestnenie intrónov.

Na viacnásobné zarovnávanie je tiež na výber veľký počet programov. Na globálne zarovnávanie niekoľkých celých sekvencií sa používajú napríklad programy CLUSTAL-W (Higgins et al., 1996) a MUSCLE (Edgar, 2004). Program TBA (Blanchette et al., 2004) hľadá viacnásobné celogenómové zarovnania.

2.9 Zhrnutie

V tejto kapitole sme sa zaoberali hľadaním podobností medzi sekvenciami DNA a proteínov. Ide o dôležitý problém, ktorý potrebujeme riešiť pri štúdiu evolúcie, ale aj pri určovaní funkcie a štruktúry nových sekvencií na základe podobnosti so známymi sekvenciami. Ukázali sme si, ako je možné problém presne definovať pomocou skórovacieho systému. Dve sekvencie vieme zarovnať v kvadratickom čase pomocou dynamického programovania. Na zrýchlenie sa však používajú heuristické algoritmy. Tie sa používajú aj na zarovnanie viacerých sekvencií, lebo ide o NP-ťažký problém, ktorý nevieme efektívne riešiť.

V tejto kapitole sme niekoľkokrát použili pravdepodobnostné modely, pričom sme uvažovali buď modely zarovnaní, v ktorých sú náhodne rozmiestnené zhody a nezhody, ale aj úplne náhodné nesúvisiace sekvencie. Takéto modely sme použili na zostavovanie skórovacích matíc, určovanie štatistickej významnosti nájdených zarovnaní, ale aj na štúdium vlastností heuristických algoritmov.

Ďalšie informácie o zarovnávaní sekvencií možno nájsť napríklad v kapitole 2 učebnice Durbin et al. (1998).

Kapitola 3

Modely evolúcie a fylogenetické stromy

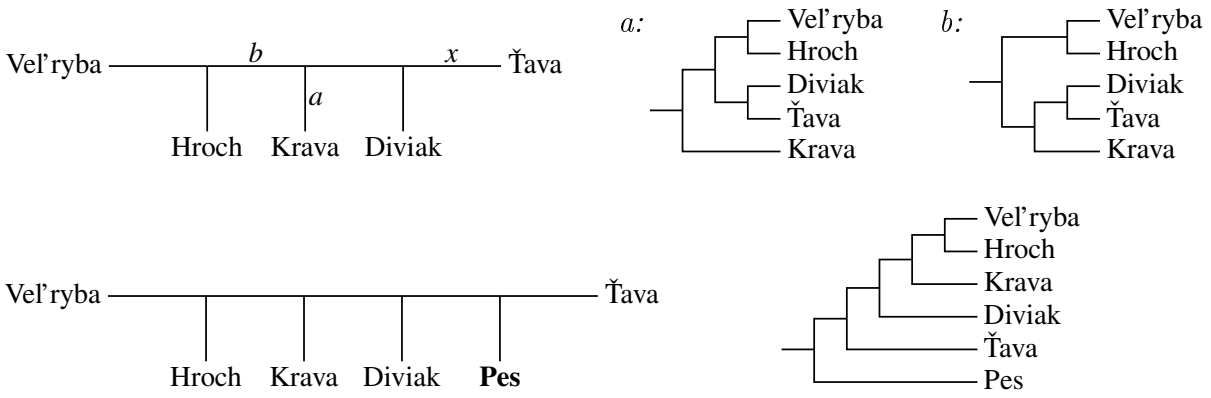
Porovnávaním homologických sekvencií z viacerých organizmov môžeme usudzovať o ich evolučnej histórii. Základnou úlohou je zostavovanie fylogenetických stromov pre skupinu organizmov. Príklad takých stromov vidíme na obrázku 3.1. Na základe morfologických znakov a fosílnych nálezov vedci dlho verili, že skutočnej evolučnej histórii zodpovedá strom naľavo, v ktorom sa najskôr oddelila skupina obsahujúca veľryby a potom sa oddelili skupina obsahujúca ťavy a kravy od skupiny obsahujúcej diviaky a hrochy. Na základe informácie z DNA sekvencií sa však dnes verí, že správny je strom napravo, v ktorom najbližšími príbuznými veľrýb sú hrochy (O'Leary and Gatesy, 2008).

Pri zostavovaní fylogenetického stromu väčšinou vychádzame z viacnásobného zarovnania niekoľkých sekvencií z rôznych druhov a snažíme sa nájsť strom, ktorý by čo najlepšie zodpovedal týmto sekvenciám. V tejto kapitole si postupne ukážeme tri rôzne kritériá, ako takýto najlepší strom zadefinovať, ako aj algoritmy na jeho hľadanie.

3.1 Fylogenetické stromy

Na popis *fylogenetického stromu* (*phylogenetic tree*) sa používajú termíny popisujúce stromy v informatike, ale aj odlišné termíny s pôvodom v biológii. Strom tak pozostáva z *vrcholov* (*vertices*) alebo *uzlov* (*nodes*) pospájaných hranami (*edges*) alebo *vetvami* (*branches*). *Koreň* (*root*) stromu zodpovedá poslednému spoločnému predkovi organizmov reprezentovaných v strome. *Listy* (*leaves*) sú vrcholy, ktoré nemajú deti, ide z nich iba jedna hrana do rodiča. Listy väčšinou zodpovedajú žijúcim biologickým druhom. *Vnútorné vrcholy* (*internal vertices*) majú väčšinou dve deti, môžu ich však mať aj viac. Tieto vrcholy reprezentujú vyhynutých predkov dnešných organizmov. Každý vnútorný vrchol

Obr. 3.1: Dva možné scenáre evolučnej histórie. Strom vľavo bol zostrojený na základe morfologických znakov a fosílnych nálezov, strom vpravo na základe DNA sekvencií.



Obr. 3.2: Zakorenené a nezakorenené stromy. V hornom riadku je vľavo nezakorenený strom. Ak ho zakoreníme na hrane označenej písmenom x, dostaneme strom na obrázku 3.1 vpravo. Môžeme ho však zakoreniť na hociktorej inej hrane, dva príklady sú uvedené na obrázku (zakorenenie na hranách označených písmenami a a b). V dolnom riadku je strom s pridaným listom pre psa (outgroup), ktorý nám pomôže strom zakoreniť ako vidíme vpravo.

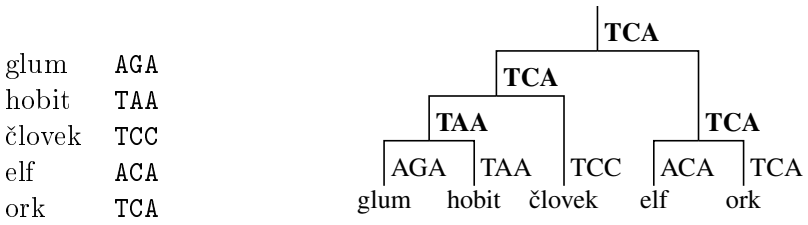
stromu reprezentuje *speciáciu* (*speciation*), t.j. udalosť, keď sa spoločná ancestrálna populácia rozdelila na dve alebo viac častí, z ktorých sa neskôr vyvinuli rôzne biologické druhy. Jednotlivým hranám fylogenetického stromu často priraďujeme určitú dĺžku, ktorá zodpovedá evolučnému času medzi dvoma speciáciami, alebo množstvu mutácií, ktoré v sekvencii na tejto hrane nastali.

Ani jedna z metód na zostavovanie stromov, ktoré budeme popisovať v tejto kapitole, žiaľ nevie určiť, kde v strome má byť umiestnený koreň. Namiesto *zakorenených stromov* (*rooted trees*), ktoré prirodzene zodpovedajú našej predstave o evolúcii, tak dostávame *nezakorenené stromy* (*unrooted trees*). Nezakorenený strom môžeme zakoreniť veľa spôsobmi: vezmeme ľubovoľnú hranu (vetvu) stromu a do jej stredu pridáme nový vrchol, ktorý sa stane novým koreňom (obr. 3.2). Nakoľko však len zo zarovnania nevieme väčšinou spoľahlivo určiť, na ktorej vetve má koreň byť, pomáhame si pridaním ďalšej sekvencie, o ktorej vieme, že nepatrí do nami študovanej skupiny (*outgroup*). Napríklad, keď zostavujeme fylogenetický strom viacerých cicavcov, môžeme zvoliť ako outgroup sekvenciu nejakého vtáka. Potom miesto, kde sa táto sekvencia pripája k stromu cicavcov, bude koreňom stromu (pozri príklad na obrázku 3.2).

Na počítačové spracovanie sa fylogenetické stromy väčšinou ukladajú v takzvanom Newickovom formáte, ktorý každý vnútorný vrchol zapíše ako pár zátvoriek, vnútri ktorého sú jeho deti oddelené čiarkami. Ak je dieťa list, uvedie sa jednoducho názov druhu, ak je to vnútroný vrchol uvedie sa ďalšia zátvorka, v ktorej vnútri budú podľa potreby ďalšie zátvorky a listy. Napríklad strom na obrázku 3.1 vľavo by sme zapísali ako

$$(\text{veľryba}, ((\text{krava}, \text{ťava}), (\text{hroch}, \text{diviak}))).$$

Keď vykresľujeme fylogenetické stromy, deti každého vrcholu musíme vykresliť v určitom poradí, teda jedno dieťa dáme naľavo a jedno napravo (alebo hore a dole). Toto poradie je však úplne ľubovoľné a nemá vplyv na význam fylogenetického stromu. Ak chceme zistiť, či sú dva stromy rovnaké, musíme skúsiť povymieňať deti vnútorných listov tak, aby listy boli v rovnakom poradí.



Obr. 3.3: Príklad stromu s ancestrálnymi sekvenciami. V listoch sú známe sekvencie, vo vnútroných vrcholoch sú dopočítané ancestrálne sekvencie. Celkový počet mutácií v celom strome je 5.

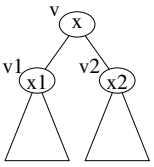
3.2 Metóda maximálnej úspornosti

Metóda *maximálnej úspornosti* (*maximum parsimony*) sa používa hlavne na blízke druhy, ktorých sekvencie sa len pomerne málo navzájom líšia. Jej cieľom je nájsť evolučnú históriu, ktorá vysvetľuje dnešné sekvencie s minimálnym počtom mutácií. Na vstupe máme viacnásobné zarovnanie sekvencií z niekoľkých organizmov. Pre jednoduchosť budeme predpokladať, že zarovnanie neobsahuje pomlčky a teda všetky vstupné sekvencie majú rovnakú dĺžku n. Cieľom je nájsť fylogenetický strom, ktorý bude mať dané organizmy v listoch a tiež nájsť predpokladanú ancestrálnu sekvenciu dĺžky n pre každý vnútorný vrchol stromu. Zo všetkých možných stromov a ancestrálnych sekvencií vyberieme také, ktoré minimalizujú počet mutácií v histórii. Tento počet vieme zrátať veľmi jednoducho: prejdeme cez všetky hrany stromu a spočítame počet miest, na ktorých sa sekvencie na koncoch hrany líšia. To je najmenší počet mutácií, ktorý sa na tejto hrane musel udiať (obr. 3.3).

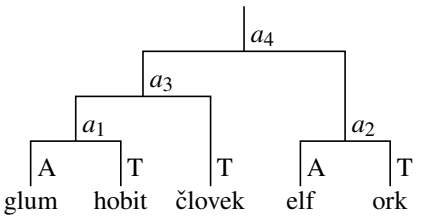
Takto sme dostali dobre definovaný informatický problém, lebo máme jasne definovaný vzťah medzi vstupom (zarovnané sekvencie) a výstupom (strom, ancestrálne sekvencie). Žiaľ, tento problém nevieme efektívne riešiť, lebo patrí tiež medzi NP-ťažké problémy. Ako si však ukážeme nižšie, vieme aspoň pomerne rýchlo spočítať úspornosť určitého stromu.

3.2.1 Výpočet úspornosti stromu

V tomto jednoduchšom probléme máme na vstupe daný strom a zarovnané sekvencie v listoch a chceme nájsť ancestrálne sekvencie tak, aby súčet mutácií na hranách bol minimálny. Ukážeme si Sankoffov algoritmus založený na dynamickom programovaní. Algoritmus uvažuje každý stĺpec zarovnania zvlášť, lebo voľba ancestrálnych sekvencií v jednom stĺpci neovplyvňuje voľbu pre iné stĺpce. Máme teda daný fylogenetický strom a bázu (alebo aminokyselinu) v každom liste a chceme vypočítať ancestrálnu bázu vo vnútorných vrcholoch. Podobne ako pri zarovnávaní, budeme vypĺňať tabuľku A. Riadky tabuľky budú zodpovedať vrcholom (uzlom) stromu a stĺpce rôznym bázam alebo aminokyselinám. Pre vrchol v a bázu x bude tabuľka obsahovať hodnotu $A[v, x]$, ktorá vyjadruje počet mutácií v podstrome s koreňom vo vrchole v, ak vo vrchole v je báza x. Tak, ako sme pri zarovnávaní uvažovali prvých niekoľko báz zo vstupnej sekvencie, v tomto algoritme uvažujeme podstrom, t.j. časť stromu, ktorú by sme dostali, ak by sme odstrihli vrchol v od jeho rodiča a uvažovali iba tie vrcholy, ktoré sú potomkami v.



Obr. 3.4: Označenie vrcholov a báz pre vzorec dynamického programovania.



Vrchol	A	C	G	T
glum	0	–	–	–
hobit	–	–	–	0
človek	–	–	–	0
elf	0	–	–	–
ork	–	–	–	0
a_1	1	2	2	1
a_2	1	2	2	1
a_3	2	3	3	1
a_4	3	4	4	**2**

Obr. 3.5: Príklad výpočtu úspornosti pre jeden stĺpec zarovnania Sankoffovým algoritmom. Vľavo je strom s bázami v listoch a s označením vnútorných vrcholov. Vpravo je tabuľka dynamického programovania. V riadku pre koreň a_4 je najmenšie číslo 2, čo určuje celkový počet mutácií.

Tabuľku budeme vypĺňať od menších podstromov k väčším. Najskôr vyplníme hodnoty $A[v, x]$ pre listy. Ak je v liste v naozaj báza x, $N[v, x]$ bude nula, lebo celý podstrom je tvorený iba vrcholom v a nedošlo v ňom teda k žiadnej mutácii. Ak je v liste v nejaká báza y líšiaca sa od x, $A[v, x]$ môžeme nastaviť na hodnotu nekonečno, lebo nemôžeme v tomto vrchole mať naraz aj x aj y. Poďme teraz spočítať $A[v, x]$ pre vnútorný vrchol v s dvoma deťmi v_1 a v_2. Nech x_1 a x_2 sú bázy pre tieto deti v optimálnom riešení (obr 3.4). Potom celkový počet mutácií v podstrome bude pozostávať z mutácií v podstrome s koreňom v_1 (tých je $A[v_1, x_1]$), z mutácií v podstrome s koreňom v_2 (tých je $A[v_2, x_2]$) a z mutácií na hranách od v k v_1 a v_2. Ak sa napríklad x_1 rovná x, tak nemuselo dôjsť k žiadnej mutácii, pre túto hranu teda pripočítame hodnotu nula. Naopak, ak sa x_1 líši od x, pripočítame jednu mutáciu. V skutočnosti však nevieme, aké sú v optimálnom riešení bázy x_1 a x_2 (podobne ako pri zarovnávaní sme nevedeli, aký je posledný stĺpec zarovnania). Preto vyskúšame všetky možnosti hodnôt x_1 a x_2 a zvolíme tie, ktoré vedú k najmenšej celkovej hodnote $A[v, x]$. Dostávame tak nasledujúci vzťah pre výpočet $A[v, x]$:

$$A[v,x] = \left(\min_{x_1 \in \{A,C,G,T\}} A[v_1, x_1] + m(x_1, x)\right) + \left(\min_{x_2 \in \{A,C,G,T\}} A[v_2, x_2] + m(x_2, x)\right)$$

Ako $m(x, y)$ sme označili počet mutácií potrebný na hrane medzi bázou x a y, teda $m(x, y)$ je 0, ak $x = y$ a 1, ak $x \neq y$. Všimnite si, že môžeme zvlášť nájsť najlepšiu hodnotu pre x_1 a zvlášť pre x_2 (a vzťah môžeme ľahko rozšíriť aj na prípad vrcholu s viac ako dvoma deťmi). Príklad tabuľky dynamického programovania je na obrázku 3.5.

Pomocou tohto vzťahu teda vypočítame hodnoty $A[v, x]$ pre všetky vnútorné vrcholy v smerom od listov ku koreňu. V koreni r potom zvolíme tú bázu x, pre ktorú je hodnota

$A[r, x]$ najmenšia. Táto hodnota bude vyjadrovať najmenší počet mutácií potrebný na vysvetlenie daných báz v listoch nášho stromu. Ak chceme dopočítať aj hodnoty ancestrálnych báz v jednotlivých vnútorných vrcholoch, pamätáme si aj hodnoty $B[v, x]$, ktoré udávajú kombináciu báz x_1 a x_2, ktorá viedla pri výpočte $A[v, x]$ k najmenšiemu počtu mutácií. Tieto hodnoty nám potom podobne ako spätné šípky pri zarovnávaní umožnia spočítať ancestrálne bázy vo všetkých vrcholoch smerom od koreňa dolu.

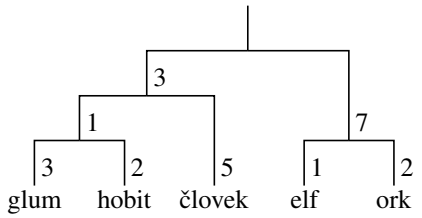
Čas na výpočet jedného políčka $A[v, x]$ je $O(\sigma)$, kde σ je počet rôznych báz, t.j. v prípade DNA $\sigma = 4$ a v prípade proteínov $\sigma = 20$. Ak má strom m listov, má $m - 1$ vnútorných vrcholov (prípadne aj menej, ak sú niektoré vrcholy stupňa viac ako 2). Tabuľka má teda $2m - 1$ riadkov a σ stĺpcov, celkový čas výpočtu je $O(m\sigma^2)$. Ak celé zarovnanie má n stĺpcov, čas výpočtu úspornosti daného stromu je $O(mn\sigma^2)$.

3.2.2 Hľadanie najúspornejšieho stromu

Pôvodne sme však nechceli rátať úspornosť nejakého daného stromu, ale nájsť najúspornejší strom pre dané zarovnanie. Jedna možnosť je vyskúšať všetky možné stromy a pre každý zrátať jeho úspornosť. Pre m sekvencií existuje $1 \cdot 3 \cdot 5 \cdots (2m - 5)$ rôznych nezakorenených fylogenetických stromov (takýto súčin sa zvykne označovať aj symbolom dvojitého faktoriálu $(2m-5)!!$). Pri počítaní úspornosti stromu dostaneme rovnakú odpoveď pri všetkých možných polohách koreňa v rámci nezakoreneného stromu a teda stačí vyskúšať jednu z nich. Ak m je malé číslo, ide o pomerne dobrý spôsob, ako najúspornejší strom nájsť. Napríklad pre 5 sekvencií potrebujeme vyskúšať iba 15 stromov. Tento počet však exponenciálne rýchlo rastie a už pre 10 sekvencií musíme skúšať asi dva milióny stromov a pre 20 sekvencií až $2 \cdot 10^{20}$ stromov.

Nakoľko ide o NP-ťažký problém, používajú sa pre väčší počet sekvencií heuristiky, ktoré začnú z nejakého stromu (napríklad zo stromu nájdeného metódou spájania susedov, ktorú popíšeme nižšie) a spočítajú jeho úspornosť Sankoffovým algoritmom. Potom skúsia tento strom mierne zmeniť a spočítajú, či zmena viedla k zlepšeniu. Ak áno, pôvodný strom sa nahradí zmeneným, ak nie, necháme si pôvodný strom. Takto postupne strom vylepšujeme, až kým nenájdeme strom, ktorý sa už nedá zlepšiť žiadnou jednotlivou operáciou z množiny povolených zmien, ktoré uvažujeme. Takéto algoritmy, ktoré sa snažia drobnými lokálnymi zmenami prísť k čo najlepšiemu výsledku, sa v informatike nazývajú *horolezecké algoritmy* (*hill climbing*). Množinu všetkých stromov si môžeme predstaviť ako pomyselný terén a čím je strom lepší (t.j. čím menej potrebuje mutácií), tým má v našom teréne vyššiu nadmorskú výšku. Chceli by sme nájsť bod s najvyššou nadmorskou výškou (teda strom s najmenej mutáciami). V našom algoritme začneme z určitého bodu v našom teréne a presunieme sa do susedného bodu len ak má vyššiu nadmorskú výšku. Takto postupne doputujeme až na vrchol nejakého kopca v našom teréne, t.j. do bodu, okolo ktorého sú len body s nižšou nadmorskou výškou. Nemusí to však vždy byť celkovo najvyšší bod – ak by sme prekonali nejaké údolia, možno by sme sa dostali aj na vrchol vyššieho kopca, to však náš algoritmus neumožňuje. V horolezeckých algoritmoch sa niekedy opakuje celý výpočet viackrát z rôznych počiatočných bodov, aby sa zvýšila šanca, že aspoň jeden začiatok povedie k najvyššiemu vrcholu.

Zostáva nám len určiť, ktoré fylogenetické stromy budú tvoriť okolie práve skúmaného stromu v horolezeckom algoritme. Jedna možnosť je použiť stratégiu SPR (*subtree pruning and regraft*). Tá funguje tak, že prerežeme jednu hranu pôvodného stromu, čím sa nám



Vrchol	glum	hobit	človek	elf	ork
glum	0	5	9	15	16
hobit	5	0	8	14	15
človek	9	8	0	16	17
elf	15	14	16	0	3
ork	16	15	17	3	0

Obr. 3.6: Príklad stromu s dĺžkami hrán a aditívnej tabuľky, ktorá tomuto stromu zodpovedá. Napríklad vzdialenosť hobita a človeka je 8, čo je súčet vzdialenosti 2 medzi hobitom a predkom hobita a gluma, vzdialenosti 1 medzi týmto predkom a predkom hobita, gluma a človeka a vzdialenosti 5 medzi týmto predkom a človekom. Ak máme tabuľku vpravo na vstupe, metóda spájania susedov zrekonštruuje strom naľavo, až na to, že nebude vedieť určiť polohu koreňa.

rozpadne na dva podstromy. V jednom podstrome si opäť zvolíme jednu hranu, v jej strede vyrobíme nový vrchol a tam pripojíme druhý podstrom. Pre strom s m listami máme teda kvadratický počet stromov v jeho okolí, lebo dve hrany zúčastňujúce sa oprácii SPR si môžeme voliť ľubovoľne.

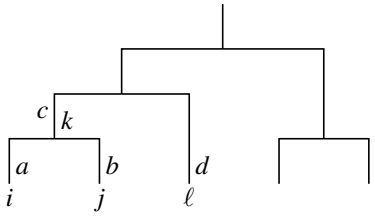
3.3 Metóda spájania susedov

Na rozdiel od metódy maximálnej úspornosti, *metóda spájania susedov* (*neighbor joining*) nedostáva na vstupe celé viacnásobné zarovnanie m sekvencií, ale len tabuľku vzdialeností $m \times m$ medzi jednotlivými sekvenciami. Túto tabuľku si označme ako D, pričom $D_{i,j}$ je vzdialenosť sekvencií i a j. Zjednodušene si môžeme predstaviť, že $D_{i,j}$ je počet rozdielov v zarovnaní sekvencií i a j, aj keď neskôr si ukážeme, že na výpočet tabuľky je lepšie použiť pravdepodobnostný model evolúcie.

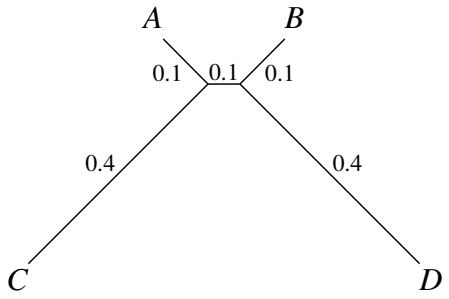
Ak nám niekto dá strom, kde každá hrana má dĺžku, vieme veľmi jednoducho spočítať vzdialenosť medzi každými dvoma listami – totiž, medzi každými dvoma listami vedie práve jedna cesta; stačí spočítať dĺžky hrán na nej. Inými slovami, pre daný strom T vieme jednoducho spočítať tabuľku $D(T)$ vzdialeností medzi listami. Metóda spájania susedov sa snaží o presný opak: pre danú tabuľku vzdialeností D nájsť strom T s dĺžkami hrán tak, aby $D(T) = D$. Zrejme nie ku každej tabuľke D existuje nejaký strom. Tabuľky, pre ktoré existuje, voláme *aditívne* a metóda spájania susedov dokáže vyhovujúci strom vždy nájsť (pozri príklad na obrázku 3.6).

Predpokladajme teraz, že naša tabuľka D je aditívna. Metóda spájania susedov postupuje od listov smerom ku koreňu stromu, pričom v každom kroku spojí nejaké dva listy, alebo už vybudované menšie stromy. Spája vždy také listy, o ktorých na základe tabuľky D vieme s určitosťou povedať, že musia mať spoločného otca. Ak sa v určitom kroku spojili listy i a j a vytvoril sa ich spoločný otec k, riadky a stĺpce pre i a j vyškrtneme z tabuľky a pridáme nový riadok a stĺpec pre vrchol k, s ktorým budeme pracovať, ako keby to bol list. Potrebujeme dopočítať vzdialenosti $D_{k,\ell}$ od k ku každému ďalšiemu listu ℓ. To dosiahneme nasledujúcim vzorcom, ktorého grafické vysvetlenie nájdete na obrázku 3.7:

$$D_{k,\ell} = \frac{D_{i,\ell} + D_{j,\ell} - D_{i,j}}{2}.$$



Obr. 3.7: Po spojení listov i a j v metóde spájania susedov ich nahradíme ich spoločným predkom k a musíme spočítať vzdialenosť $D_{k,\ell}$ ku každému inému listu ℓ. Ak si vezmeme $D_{i,\ell} + D_{j,\ell}$, tak hrany na ceste z k do ℓ (t.j. v tomto prípade hrany s dĺžkami c a d) započítame dvakrát a hrany medzi i a k a j a k iba raz. Ak teda odčíteme $D_{i,j}$, dostávame, že $D_{i,\ell}+D_{j,\ell}-D_{i,j}$ je dvojnásobok $D_{k,\ell}$ (v našom prípade $(a+c+d)+(b+c+d)-(a+b) = 2(c+d)$, čo je naozaj $2D_{k,\ell}$).



Obr. 3.8: Ak by sme pri spájaní listov spojili vždy dva s najmenšou vzdialenosťou, mohli by sme dostať zlú odpoveď.

V jednom kroku nám tak počet riadkov aj stĺpcov tabuľky klesne o jedna a po $m-1$ krokoch pospájame všetky pôvodné listy do jedného výsledného stromu.

Zostáva nám určiť, ktoré dva listy i a j je možné spojiť. Intuitívne by sa mohlo zdať vhodné spájať vždy dva najbližšie listy, t.j. nájsť v tabuľke najmenšiu hodnotu a spojiť listy zodpovedajúce jej riadku a stĺpcu. Pre niektoré tabuľky by sme tak však dostali nesprávny výsledok (pozri obrázok 3.8). Preto sa tabuľka D najskôr prepočíta na novú tabuľku L a spojíme také listy i a j, pre ktoré je $L_{i,j}$ najmenšie. Pre každý list i si najskôr spočítame súčet vzdialeností r_i do všetkých ostatných listov, teda $r_i = \sum_{k \neq i} D_{i,k}$. Hodnota $L_{i,j}$ potom bude

$$L_{i,j} = (m-2)D_{i,j} - r_i - r_j.$$

Hlavnou výhodou metódy spájania susedov je jej rýchlosť. Na vytvorenie tabuľky D ani nepotrebujeme zostavovať viacnásobné zarovnanie, stačia nám párové zarovnania medzi jednotlivými dvojicami. Samotný aloritmus potom beží v čase $O(m^3)$, lebo robíme $m-1$ krokov a v každom kroku prepočítame tabuľku $L_{i,j}$ v čase $O(m^2)$. V praxi tabuľka D väčšinou nie je aditívna, metóda spájania susedov teda nájde strom, v ktorom vzdialenosti nebudú sedieť s D.

3.4 Pravdepodobnostné modely evolúcie

V tejto časti si ukážeme jednoduché pravdepodobnostné modely evolúcie, ktoré sa dajú použiť na získanie presnejšej tabuľky D pre metódu spájania susedov a tvoria aj základ

metódy maximálnej vierohodnosti, ktorú si popíšeme nižšie. Opäť budeme uvažovať iba proces substitúcie, t.j. zmeny jednej bázy na inú, a nie procesy inzercie a delécie, ktoré vedú k medzerám v zarovnaniach sekvencií.

Predstavme si, že pozorujeme jednu bázu DNA v čase. Pôvodne to mohla byť napríklad báza `T`. V čase t_1 sa zmení napríklad na `C` a neskôr v čase t_2 na `A`. Tento proces mutácií je do značnej miery náhodný, takže nevieme vopred odhadnúť kedy a na akú bázu bude daná báza mutovať. Tento proces budeme modelovať pravdepodobnostným modelom, ktorý sa snaží túto náhodnosť zachytiť, aj keď samozrejme s určitými zjednodušeniami.

Modely substitúcií väčšinou uvažujú, že každá báza sa mení nezávisle. Používajú spojitý čas, takže čas medzi dvoma substitúciami tej istej bázy môže byť ľubovoľné kladné reálne číslo. Nech X_t je báza na určitej pozícii v sekvencii v čase t. Model nám definuje pravdepodobnosti typu $\Pr(X_{t+\Delta} = \texttt{C} \mid X_t = \texttt{A})$, t.j., aká je šanca, že ak začneme v čase t s bázou `A`, tak po uplynutí času Δ budeme mať na tej istej pozícii bázu `C`. Táto pravdepodobnosť zahŕňa prípad, že `A` zmutuje na `C` niekedy medzi časom t a $t + \Delta$, ale aj prípady, keď v tomto časovom intervale dôjde k viacerým mutáciám, napríklad z `A` na `T` a potom na `C`.

Najjednoduchší model substitúcií je *Jukes-Cantorov model*, v ktorom všetky substitúcie nastávajú rovnako rýchlo, takže pravdepodobnosť zmeny napríklad z `A` na `G` je rovnaká ako z `T` na `A`. V tomto modeli dostávame nasledujúci vzťah pre pravdepodobnosť zmeny z bázy x na inú bázu $y \neq x$:

$$\Pr(X_{t+\Delta} = x \mid X_t = y) = \frac{1}{4}(1 - e^{-\frac{4}{3}\alpha\Delta}).$$

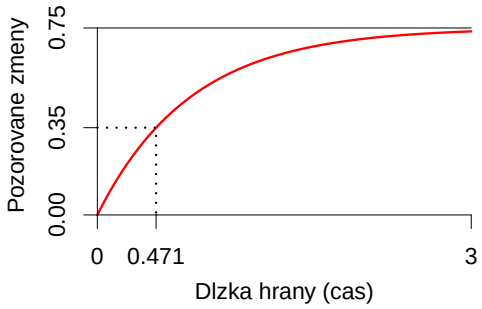
V tomto vzorci je α rýchlosť substitúcií, teda čím vyššie α, tým viac substitúcií nastane v priemere za jednotku času. Ak začneme z bázy `A`, tak po čase Δ budeme mať jednu z báz `A`, `C`, `G` alebo `T`, takže súčet ich pravdepodobností musí byť jedna. Z toho vieme odvodiť pravdepodobnosť $\Pr(X_{t+\Delta} = x \mid X_t = x)$:

$$\Pr(X_{t+\Delta} = x \mid X_t = x) = \frac{1}{4}(1 + 3e^{-\frac{4}{3}\alpha\Delta}).$$

Táto pravdepodobnosť zahŕňa prípad, že sa `A` vôbec nezmenilo, alebo že prebehlo viacero substitúcií, pričom posledná z nich zmenila bázu späť na `A`. V týchto vzorcoch sa vyskytuje člen $\alpha\Delta$, ktorý je súčinom rýchlosti substitúcií α a času Δ vyjadreného vo vhodných jednotkách. Keďže čas je ťažké kalibrovať na reálne jednotky, zvolíme väčšinou α tak, aby priemerný počet substitúcií jednej bázy za jednotku času bol jedna, čo v Jukes-Cantorovom modeli zodpovedá $\alpha = 1$ a teda meriame čas nie v reálnych fyzikálnych jednotkách, ale v substitúciách na bázu.

V praxi sa často používajú aj zložitejšie modely substitúcií, v ktorých uvažujeme napríklad to, že rôzne bázy alebo amino kyseliny sa vyskytujú s rôznou frekvenciou, že rôzne substitúcie tiež nastávajú s rôznou frekvenciu a že rôzne pozície v sekvencii môžu tiež mutovať rôzne rýchlo. Takéto modely uvidíme v kapitole 5, ale záujemcov o hlbšie detaily odkazujeme najmä na kapitolu 13 v učebnici Felsenstein (2004).

Vráťme sa teraz k odhadovaniu vzdialeností medzi sekvenciami v metóde spájania susedov. Uvažujme dve sekvencie, nech f je percento báz, v ktorých sa navzájom líšia. Pre každú bázu, ktorá sa líši, musela nastať aspoň jedna substitúcia, v skutočnosti však niektoré bázy mohli zmutovať aj viackrát a teda skutočný počet substitúcií na bázu mohol



Obr. 3.9: Priemerná frekvencia pozorovaných zmien ako funkcia skutočnej evolučnej vzdialenosti v Jukes-Cantorovom modeli.

byť vyšší ako f. Veličina f teda nie je veľmi dobrým odhadom evolučnej vzdialenosti dvoch sevencií. Jukes-Cantorov model nám umožňuje spraviť korekciu, ktorá viacnásobné mutácie berie do úvahy. Pravdepodobnosť, že na danej pozícii sekvencie budeme za čas Δ pozorovať nejakú zmenu je jednoducho súčtom pravdepodobností, že ak začneme z nejakej bázy x, tak dôjde k zmene na jednu z ďalších troch báz. Pri $\alpha = 1$ dostávame pravdepodobnosť pozorovania zmeny

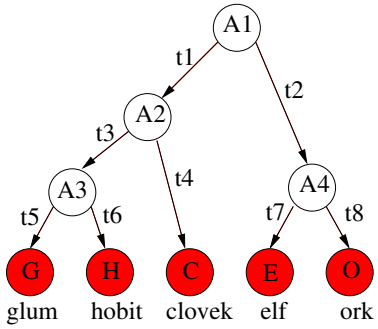
$$\Pr(X_{t+\Delta} \neq X_t) = \frac{3}{4}(1 + 3e^{-\frac{4}{3}\Delta}).$$

Na obrázku 3.9 vidíme, ako táto pravdepodobnosť rastie s časom Δ. V limite pre veľmi veľký čas sa táto pravdepodobnosť blíži k 3/4, lebo dve nesúvisiace náhodne zvolené bázy sa rovnajú s pravdepodobnosťou 1/4. My v skutočnosti vieme, že v našich dátach je frekvencia zmeny f, čo zodpovedá y-ovej osi grafu. Chceme vyjadriť hodnotu Δ, čo bude náš odhad $D_{i,j}$ evolučnej vzdialenosti medzi sekvenciami i a j. Dostávame vzťah $\Delta = -\frac{3}{4}\ln\left(1 - \frac{4}{3}f\right)$. Pre veľmi malé hodnoty f sa Δ odhadnuté týmto vzťahom príliš nelíši od f, lebo len zriedkavo nastanú viacnásobné mutácie. Ak sa ale f blíži k 3/4, odhad času rastie do nekonečna. Pre hodnoty väčšie f ako 3/4 tento vzorec na výpočet vzdialenosti nie je možné použiť.

3.5 Metóda maximálnej vierohodnosti

Metóda maximálnej vierohodnosti sa podobá na metódu maximálnej úspornosti, ale namiesto toho, aby sme hľadali strom, ktorý vyžaduje čo najmenej mutácií, hľadáme strom, ktorý sa bude zdať najvierohodnejší vhľadom na určitý model evolúcie. Berieme pritom do úvahy aj dĺžky hrán, a teda očakávame, že na dlhších hranách nastalo viac mutácií ako na kratších.

Vstupom pre metódu maximálnej vierohodnosti je teda opäť zarovnanie m sekvencií, pričom neuvažujeme medzery. Výstupom je strom, ktorý má tieto sekvencie v listoch a dĺžky hrán tohto stromu. Zo všetkých možných stromov a všetkých možných dĺžok vyberieme tie, ktoré najlepšie zodpovedajú vstupným sekvenciám v našom modeli substitúcií, napríklad v Jukes-Cantorovom modeli. To, ako dobre strom zodpovedá sekvenciám, budeme merať veličinou zvanou vierohodnosť. Aby sme mohli vierohodnosť formálne matematicky zadefinovať, musíme rozšíriť náš model evolúcie z pozorovania substitúcií jednej



Obr. 3.10: Fylogenetický strom s dĺžkami hrán $t_1, \ldots, t_8$, s ancestrálnymi bázami $A_1, \ldots, A_4$ a bázami v listoch G, H, C, E, O (ide o označenie neznámych, nie konkrétne hodnoty báz).

bázy v čase na evolúciu viacerých dlhších sekvencií. Potom si ukážeme, ako spočítať vierohodnosť pre daný strom a dĺžky hrán a nakoniec si povieme niečo o tom, ako hľadať strom s najvyššou vierohodnosťou.

3.5.1 Fylogenetický strom ako pravdepodobnostný model

Doteraz sme uvažovali proces, v ktorom sme pozorovali určitú bázu a tá sa mohla v hociktorom okamihu zmeniť na nejakú inú. Ako sme videli, substitučné modely popisujú tento proces a definujú pravdepodobnosť $\Pr(X_{t+\Delta} = y \mid X_t = x)$, že po nejakom časovom úseku Δ na mieste, kde bola báza x, bude báza y. V tejto kapitole budeme túto pravdepodobnosť označovať skráteným označením $\Pr(y \mid x, \Delta)$.

Teraz si náš pravdepodobnostný model rozšírime o pevne daný fylogentický strom so známymi dĺžkami hrán. Proces evolúcie jednej bázy sekvencie budeme modelovať náhodným procesom, v ktorom sa najskôr vygeneruje náhodná báza v koreni stromu. Na určitej hrane dĺžky t sa táto báza náhodne mení procesom popísaným vyššie, t.j. ak vo vrchole na hornom konci hrany mala hodnotu x, pravdepodobnosť, že na spodnom konci hrany bude mať hodnotu y je $\Pr(y \mid x, t)$. Vo vnútornom vrchole, ktorý spravidla zodpovedá speciácii, sa táto báza skopíruje každému z dvoch vznikajúcich organizmov a v každom sa odvtedy vyvíja nezávisle.

Tento model každej kombinácii ancestrálnych báz vo vnútorných vrcholoch a dnešných báz v listoch priradí určitú pravdepodobnosť, ktorá bude súčinom vygenerovania bázy v koreni a pravdepodobností tvaru $\Pr(y \mid x, t)$ pre jednotlivé hrany stromu. Napríklad pre strom na obrázku 3.10 dostaneme pravdepodobnosť

$$\begin{aligned}\Pr(G, H, C, E, O, A_1, \ldots, A_4) \;=\; & \Pr(A_1) \cdot \Pr(A_2 \mid A_1, t_1) \cdot \Pr(A_4 \mid A_1, t_2) \\ & \cdot \Pr(A_3 \mid A_2, t_3) \cdot \Pr(C \mid A_2, t_4) \cdot \Pr(G \mid A_3, t_5) \\ & \cdot \Pr(H \mid A_3, t_6) \cdot \Pr(E \mid A_4, t_7) \cdot \Pr(O \mid A_4, t_8)\end{aligned}$$

Jednotlivé členy tohto súčinu počítame napríklad pomocou Jukes-Cantorovho modelu, t.j.

$$\Pr(x \mid y, t) = \begin{cases} \frac{1}{4}(1 - e^{-\frac{4}{3}t}) & \text{ak } x \neq y \\ \frac{1}{4}(1 + 3e^{-\frac{4}{3}t}) & \text{ak } x = y \end{cases}$$

Hodnotu $\Pr(A_1)$ v koreni stromu môžeme nastaviť napríklad na 1/4, t.j. každá báza bude rovnako pravdepodobná.

Pri hľadaní fylogenetického stromu poznáme iba hodnoty báz v listoch, ale nepoznáme ancestrálne bázy. Zaujímalo by nás, akú pravdepodobnosť tento model prikladá určitej kombinácii báz v listoch, pričom do tejto pravdepobnosti zahrnieme všetky možné kombinácie ancestrálnych báz vo vnútorných vrcholoch. V našom konkrétnom príklade dostávame

$$\Pr(G, H, C, E, O) = \sum_{A_1,\ldots,A_4} \Pr(G, H, C, E, O, A_1, \ldots, A_4).$$

Toto je teda pravdepodobnosť, ktorú náš model priradí určitej kombinácii báz, ktoré môžu tvoriť jeden stĺpec viacnásobného zarovnania. Ak máme zarovnanie s n stĺpcami, považujeme ich za nezávislé a teda pravdepodobnosť celého zarovnania je súčinom pravdepodobností pre jednotlivé stĺpce. Doteraz sme rozprávali o pravdepodobnosti, čo je hodnota, pri ktorej uvažujeme pevne zvolený strom a hrany a meniace sa hodnoty báz. Pojem *vierohodnosť* (*likelihood*) sa vzťahuje k tomu istému vzorcu, ale považuje bázy za pevné a naopak strom a jeho dĺžky hrán za premenné. Vierohodnosť stromu je teda rovná pravdepodobnosti vstupných dát (zarovnania) ak tento strom považujeme za správny. Metóda maximálnej vierohodnosti sa teda snaží nájsť strom a dĺžky hrán tak, aby pravdepodobnosť, ktorú tento strom pripradí vstupnému zarovnaniu bola čo najvyššia.

3.5.2 Výpočet vierohodnosti stromu

Ako sme videli vyššie, pre danú kombináciu báz v listoch a ancestrálnych báz je výpočet pravdepodobnosti jednoduchým súčinom, v ktorom jednotlivé členy získame zo substitučného modelu. Ak však nepoznáme ancestrálne bázy, potrebujeme sčítať takéto pravdepodobnosti pre všetky kombinácie ancestrálnych báz. Pri m listoch máme $m-1$ vnútorných vrcholov a teda 4^{m-1} kombinácií báz, ktoré by sme museli jednu po druhej spočítať. Toto je opäť algoritmus, ktorého časová zložitosť rastie exponenciálne a teda sa nedá použiť pre väčšie počty listov.

Namiesto toho môžeme pravdepodobnosť efektívne spočítať Felsensteinovým algoritmom, čo je dynamické programovanie veľmi podobné na výpočet úspornosti stromu. Pre jeden stĺpec tento algoritmus pracuje v čase $O(m\sigma^2)$ a pre zarovnanie dĺžky n ho opakujeme n krát, teda dostávame čas $O(nm\sigma^2)$.

3.5.3 Hľadanie najvierohodnejšieho stromu

Pre daný strom a dĺžky hrán vieme zadefinovať a spočítať ich vierohodnosť. Nás však zaujíma predovšetkým hľadanie najvierohodnejšieho stromu pre dané vstupné zarovnanie sekvencií. Opäť, ako pri maximálnej úspornosti, sa ukázalo, že ide o NP-ťažký problém (Chor and Tuller, 2005). Stále však môžeme použiť horolezecké algoritmy, kde začíname z určitého stromu a postupne sa ho malými zmenami snažíme vylepšiť. Na rozdiel od maximálnej úspornosti potrebujeme nielen strom, ale aj dĺžky hrán. Tie si na začiatku tiež nejako zvolíme a postupne ich vylepšujeme technikami numerickej optimalizácie.

Tabuľka 3.1: Prehľad uvádzaných algoritmov na rekonštrukciu fylogenetických stromov.

	Zložitosť	Konzistentný	Využitie dát
Úspornosť (parsimony)	NP-ťažký	nie	celé sekvencie
Spájanie susedov (neighbor joining)	$O(m^3)$	áno	iba vzdialenosti
Vierohodnosť (likelihood)	NP-ťažký	áno	celé sekvencie

3.6 Záverečné poznámky

3.6.1 Správnosť a kozistentnosť algoritmov

V tabuľke 3.1 vidíme prehľad algoritmov na konštrukciu fylogenetických stromov, ktoré sme videli v tejto kapitole. Jedna z vlastností, ktorú môžeme pri každom algoritme študovať, je jeho konzistentnosť. Predpokladajme, že zarovnávané sekvencie boli vygenerované podľa určitého stromu a modelu substitúcií. Ak je zarovnanie krátke, nemusí obsahovať dosť dát na to, aby sme vedeli určiť správny strom. Ako však dĺžka sekvencie rastie, očakávali by sme, že algoritmus nájde správny strom. Algoritmy, ktoré túto podmienku spĺňajú, nazývame *konzistentné*. Presnejšie, algoritmus je konzistentný, ak v prípade, že dĺžka sekvencií n rastie do nekonečna, pravdepodobnosť, že algoritmus nájde správny strom sa blíži k jednej. Táto pravdepodobnosť sa počíta cez všetky zarovnania, ktoré mohol model vygenerovať: aj zarovnania, pre ktoré algoritmus spočíta nesprávny strom, majú nenulovú pravdepodobnosť, celková pravdepodobnosť všetkých takých zarovnaní však musí konvergovať k nule.

Z uvádzaných algoritmov maximálna úspornosť nie je konzistentná, ak však použijeme správny substitučný model, metóda spájania susedov aj maximálna vierohodnosť sú konzistentné algoritmy.

Konzistentnosť algoritmu je určitým kritériom jeho správnosti, v praxi nám však nezaručuje správnosť výsledných stromov, lebo reálne dáta nie sú generované zo substitučného modelu (a nemusíme ich mať dostatok). Na rozmiestnenie a typ mutácií vplýva mnoho faktorov. Niektoré z nich sa dajú zachytiť zložitejšími modelmi, ale aj tak zostanú niektoré nepokryté.

Analýza konzistentnosti však poukazuje na to, že dôležitým faktorom ovplyvňujúcim správnosť rekonštruovaných stromov je dĺžka zarovnaní. Pri dlhších sekvenciách máme viac informácie o správnom strome. Jednoduchý spôsob, ako overiť, či máme na rekonštrukciu dosť dát, je použiť techniku nazývanú *bootstrap*. Pri tejto technike náhodne vyberieme niektoré stĺpce zarovnania a zostavíme strom len podľa nich. Toto veľakrát opakujeme a dostaneme veľa stromov, každý založený na inom náhodnom výbere stĺpcov. Ak bolo dát dostatok, väčšina z týchto stromov by mala byť rovnakých ako strom zostavený zo všetkých dát. Ak však dát nie je dosť, rôznym výberom stĺpcov budeme dostávať rôzne stromy. V stromoch sa zvykne uvádzať pri jednotlivých vetvách bootstrap hodnota, ktorá vyjadruje, v koľkých stromoch sa táto vetva vyskytuje. Vetvy s vysokou boostrap hodnotou sú hodnovernejšie ako tie s nízkou. Bootstrap nám však nepomôže odhaliť, či použitý model dobre zodpovedá vstupným dátam.

Správnosť rekonštrukcie evolučných stromov závisí aj od dĺžky hrán. Pri veľmi krátkych hranách potrebujeme veľmi dlhé zarovnania, aby sme v nich našli dosť mutácií na rozlíšenie, o ktorý strom ide. Pri veľmi dlhých hranách už mohlo nastať priveľa mutá-

cií, ktoré úplne zamaskovali stopy spoločnej histórie zdieľanej niektorými druhmi. Bežný problém pri rekonštrukcii stromov je *long branch attraction*, čo je fenomén, keď máme v skutočnom strome niekoľko dlhých hrán. Metóda maximálnej úspornosti, ale do určitej miery aj ostatné algoritmy, majú tendenciu tieto hrany dať blízko k sebe a tým zrekonštruovať nesprávny strom. Niekedy sa tento problém podarí odstrániť pridaním viacerých sekvencií, ktoré sa pripájajú uprostred dlhých hrán. Také sekvencie však nie sú vždy k dispozícii.

3.6.2 Zarovnania s medzerami

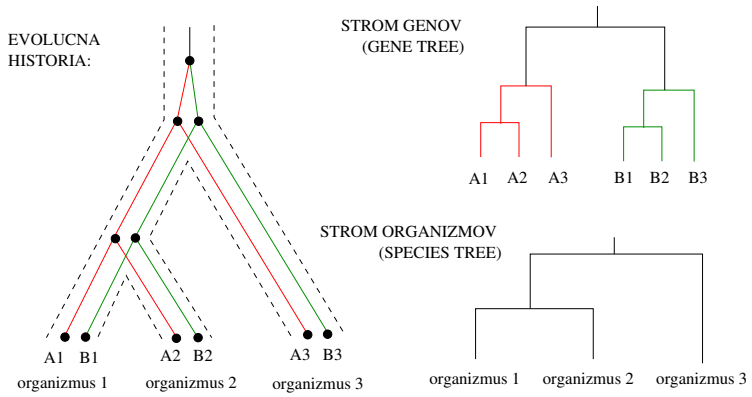
Pri popisovaní algoritmov sme predpokladali, že zarovnanie neobsahuje medzery. Skutočné zarovnania však, samozrejme, medzery obsahujú a algoritmy sa s nimi potrebujú vyrovnať. Ak je medzier iba veľmi málo a máme dosť dlhé sekvencie, môžeme jednoducho vynechať všetky stĺpce zarovnania, v ktorých sa medzery vyskytujú. Druhá možnosť je považovať medzery za chýbajúce dáta: predpokladáme, že v medzere je nejaká báza, iba nevieme, ktorá. Metóda maximálnej vierohodnosti sa s chýbajúcimi dátami veľmi ľahko vyrovná, lebo rovnako ako pri neznámych ancestrálnych bázach uvažujeme všetky jej možnosti. Podobne pri maximálnej úspornosti môžeme neznáme bázy v listoch doplniť tak, aby bol potrebný počet mutácií v celom strome čo najmenší.

Obidve tieto metódy ignorujú informáciu obsiahnutú v samotnom rozmiestnení medzier. Ďalšou možnosťou je považovať pomlčku za špeciálny typ bázy a uvažovať napríklad v substitučnom modeli rýchlosť s akou sa pomlčky menia na bázy a naopak. Tento prístup má dve nevýhody: neuvažuje, že jednou evolučnou udalosťou môže vzniknúť dlhšia medzera pozostávajúca z niekoľkých pomlčiek a tiež povoľuje evolučné histórie, v ktorých sa báza zmaže a potom opäť vloží. Takáto novovložená báza by už nemala byť považovaná za homologickú s pôvodnou bázou.

Nakoniec je možné vybudovať aj zložitejší pravdepodobnostný model, v ktorom budeme uvažovať dlhšie medzery a využiť ho či už metódou spájania susedov alebo maximálnej vierohodnosti. Takéto modely sú však pomerne zložité a menej sa využívajú v praxi.

3.6.3 Génové a druhové stromy

V tejto kapitole sme predpokladali, že jednotlivé sekvencie na vstupe pochádzajú z rôznych biologických druhov a že sa snažíme zostaviť strom, ktorý reprezentuje históriu týchto druhov. Počas evolúcie však dochádza aj k procesu duplikácie, keď sa tá istá sekvencia môže skopírovať na iné miesto v tom istom genóme a odvtedy sa vyvíjajú nezávisle. Časom sa môže v jednom genóme nahromadiť aj viacero kópií. Takéto homologické sekvencie z jedného genómu môžeme tiež zarovnať a zostaviť ich fylogenetický strom. Vnútorné vrcholy tohto stromu však nebudú zodpovedať speciácii, ale duplikácii. Ešte zložitejšia situácia nastane, ak použijeme homológy duplikovaného génu z viacerých organizmov. Niektoré vrcholy v strome odpovedajú speciáciám, iné duplikáciám (obr. 3.11) a len samotný strom neurčuje jednoznačne, ktoré vrcholy sú ktorého typu. Takýto strom sa vo všeobecnosti nazýva *génový strom* (*gene tree*), lebo zodpovedá evolučnej histórii určitého génu, alebo inej sekvencie, ktorú sme na jeho rekonštrukciu použili. Naopak *druhový strom* (*species tree*) reprezentuje evolučnú históriu príslušných druhov.



Obr. 3.11: Evolučná história duplikovaného génu.

Ak máme génový strom, v ktorom je známe označenie vrcholov ako speciačné a duplikačné, môžeme jednotlivé dvojice homologických sekvencií klasifikovať ako ortológy a paralógy. Pripomíname, že homologické sekvencie sú také, ktoré sa vyvinuli z toho istého spoločného predka. Ak vezmeme dva homológy v strome, môžeme nájsť vnútorný vrchol, ktorý je ich najbližším spoločným predkom. Ak je tento vrchol speciačný, tieto homológy nazveme *ortológmi* (*orthologs*) a ak je duplikačný, nazveme ich *paralógmi* (*paralogs*). Dve homologické sekvencie z toho istého genómu budú teda vždy paralogické, ale homológy z dvoch rôznych organizmov môžu byť buď ortológy (ako napríklad sekvencie A1 a A3 na obrázku 3.11) alebo paralógy (ako napríklad A1 a B2 na obrázku 3.11). Ak chceme zostavovať druhový strom, potrebujeme z každého genómu zvoliť sekvenciu tak, aby všetky boli navzájom ortologické. Ako sme spomínali v kapitole 2, určitým vodídkom na hľadanie ortológov je aj zachovanie poradia génov na chromozóme.

3.6.4 Zdroje dát

Pri zostavovaní druhových stromov je potrebné získať ortologické sekvencie z každého zastúpeného druhu. V princípe môžeme použiť hociktorý gén, ktorého sekvenciu máme k dispozícii zo všetkých študovaných druhov, pokiaľ sme si rozumne istí, že ide o skutočné ortológy a nie paralógy (preto je dobré vyhýbať sa génom z rodín, ktoré často podliehajú duplikácii).

Pri použití sekvencií génov kódujúcich proteíny sa musíme rohodnúť, či použijeme DNA alebo proteínové sekvencie. Proteínové sekvencie sa v evolúcii menia pomalšie, preto sú vhodnejšie pre študovanie vzdialenejších organizmov, lebo je väčšia šanca, že budú dostatočne podobné na spoľahlivé zarovnanie a fylogenetickú analýzu. Naopak, DNA sekvencie sú vhodnejšie na porovnávanie blízko príbuzných organizmov, kde sa proteínové sekvencie často líšia iba veľmi málo a neobsahujú dostatok informácie.

Ak máme k dispozícii viacero génov, môžeme ich buď zreťaziť do jedného zarovnania a použiť bežné fylogenetické metódy, alebo môžeme zostaviť strom pre každý gén zvlášť a potom hľadať konsenzus týchto stromov.

Obzvlášť populárny typ sekvencie pri štúdiu druhov, ktoré ešte nemajú osekvenovaný genóm je ribozomálna RNA (rRNA). Táto je nepostrádateľná pri syntéze proteínov v ribozómoch, a preto je veľmi dobre zachovaná aj medzi druhmi. Sekvencie rRNA z mnohých druhov sa dajú nájsť v databáze RDPII.

Ďalším často používaným typom sekvencií je mitochondriálna DNA (mtDNA). Je to krátky cirkulárny genóm uložený v mitochondriách. Napríklad u človeka má dĺžku cca 16KB. Tento genóm sa pomerne ľahko sekvenuje, lebo je krátky a nachádza sa v bunkách vo veľa kópiách. Preto je osekvenovaný pre pomerne veľa organizmov. Na rozdiel od rRNA pomerne rýchlo mutuje, takže je vhodný na porovnávanie bližších druhov alebo dokonca subpopulácií v rámci druhu.

3.7 Zhrnutie

V tejto kapitole sme si ukázali tri metódy na rekonštrukciu fylogenetických stromov z homologických sekvencií. Metóda maximálnej vierohodnosti vychádza z pravdepodobnostného modelu evolúcie, v ktorom sa jednotlivé bázy v náhodných časoch menia na iné bázy. Cieľom je pre dané viacnásobné zarovnanie nájsť strom a dĺžky hrán, ktoré vedú k maximálnej vierohodnosti, t.j. k maximálnej pravdepodobnosti daného zarovnania. Uvažuje pritom všetky možné scenáre mutácií a ancestrálnych báz. Popísali sme iba Jukes-Cantorov model, ale v praxi sa spravidla používajú zložitejšie modely s viacerými parametrami, ktoré sa optimalizujú spolu so stromom a dĺžkami hrán. Nevýhodou tejto metódy je jej časová náročnosť, ktorá neumožňuje nájdenie globálneho optima, ale iba lokálne prehľadávanie priestoru stromov.

Pravdepodobnostné modely sa tiež používajú na odhad vzdialeností medzi dvojicami sekvencií a z nich potom vieme zostaviť fylogenetický strom metódou spájania susedov. Táto metóda je rýchla a štatisticky konzistentná, ale nevyužíva všetku informáciu obsiahnutú v sekvenciách.

Napokon metóda maximálnej úspornosti je jednoduchá a dáva dobré výsledky pre pomerne blízke druhy, kde nie je priveľa viacnásobných mutácií. Aj táto metóda sa však používa s lokálnym heuristickým prehľadávaním priestoru stromov.

Kapitola 4

Hľadanie génov

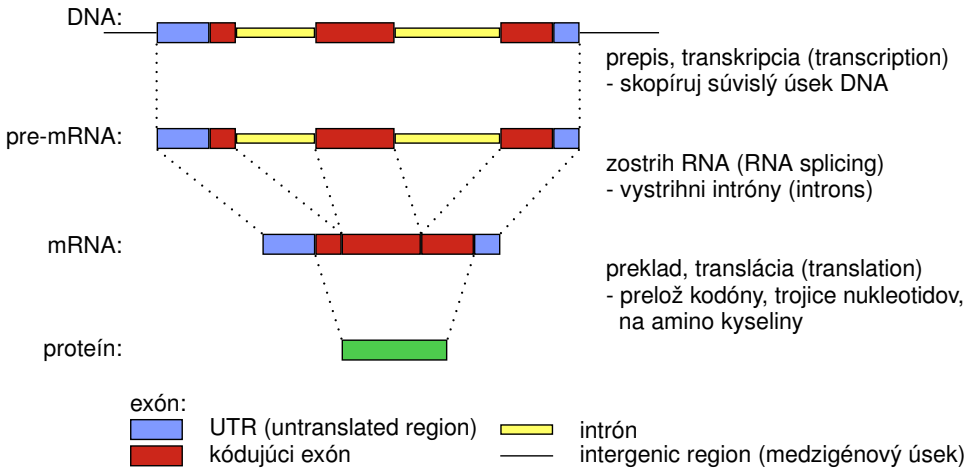
Doteraz sme sa zaoberali získavaním genomických sekvencií a ich využitím na štúdium podobností a rozdielov medzi nimi a možných evolučných histórií. Informatické metódy sú tiež veľmi dôležité pri anotácii genómov, teda určovaní významu a funkcie jednotlivých oblastí genómu. V tejto kapitole sa sústredíme na hľadanie génov kódujúcich proteíny (*protein coding genes*). Pri hľadaní génov máme ako vstup DNA sekvenciu a chceme v nej nájsť všetky oblasti, ktoré kódujú proteíny. Na základe genetického kódu potom vieme určiť poradie aminokyselín v týchto proteínov, čo je predpoklad, pre ďalšie štúdium ich funkcie.

Okrem génov kódujúcich proteíny sa pri anotácii genómov snažíme objaviť aj iné funkčné prvky, napríklad RNA gény a signály pre reguláciu transcripcie. Okrem toho v sekvencii hľadáme aj oblasti, ktoré nemajú jasne definovanú funkciu, napríklad pseudogény (*pseudogenes*) a sekvenčné opakovania (*sequence repeats*). Pseudogény sú pozostatky bývalých génov, ktoré boli znefunkčnené mutáciami, ale ktoré ešte nesú podobnosť k príbuzným funkčným génom. Sekvenčné opakovania sú rôzne typy sekvencií, ktoré sa v genóme vyskytujú vo veľkom počte podobných kópií, či už zhromaždených v jednej oblasti, alebo roztrúsených po celom genóme. Pseudogény aj opakovania sú zaujímavé pri štúdiu evolúcie, ale môžu spôsobovať problémy. Napríklad programy na hľadanie génov môžu pseudogény označiť za funkčné gény a programy na hľadanie homológov môžu nájsť veľké množstvo podobností medzi sekvenčnými opakovaniami, ktoré nás pri mnohých druhoch analýz príliš nezaujímajú. Preto je dobré tieto druhy sekvencií vopred nájsť a v niektorých prípadoch aj vylúčiť z ďalšieho spracovania.

V tejto kapitole sa však budeme zaoberať špecificky hľadaním génov kódujúcich proteíny, pričom sa sústredíme hlavne na eukaryotické organizmy, v ktorých je táto úloha ťažšia, než v prokaryotoch. Najskôr popíšeme typické vlastosti génov a potom sa budeme zaoberať skrytými Markovovými modelmi, čo sú pravdepodobnostné modely využívané na hľadanie génov, ale aj iné bioinformatické problémy.

4.1 Vlastnosti eukaryotických génov

Pri expresii génov sa určitá oblasť najskôr transkribuje do RNA, táto RNA sa ďalej spracováva a vzniknutá mRNA po prechode z jadra do cytoplazmy slúži ako predloha na tvorbu proteínov, pričom vždy trojica báz (kodón) kóduje jednu aminokyselinu proteínu. Pri hľadaní génov potrebujeme podrobnejšie popísať spracovanie RNA, a to najmä proces

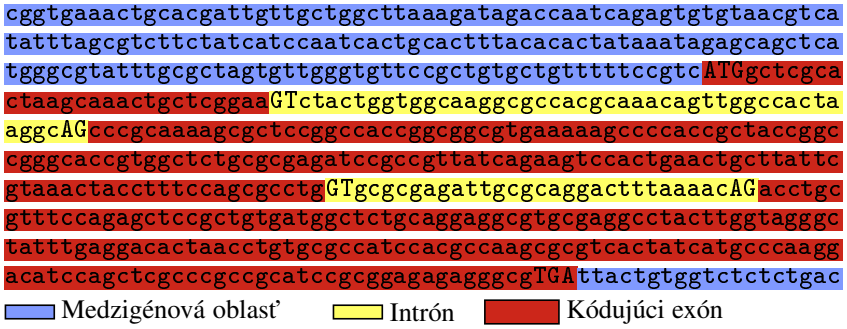


Obr. 4.1: Proces expresie génu začína transcripciou z DNA do RNA, z RNA sú vystrihnuté intróny a potom je prekladaná do proteínu.

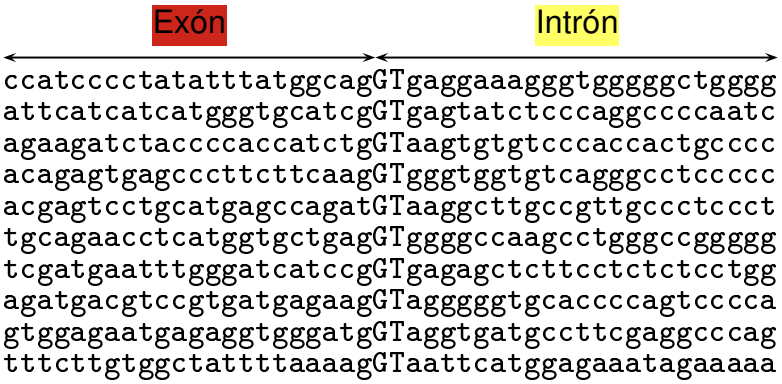
nazývaný RNA zostrih (*RNA splicing*). Pri tomto procese sa vystrihnú niektoré úseky RNA a zvyšné časti sa „zlepia“ spolu v pôvodnom poradí (obr. 4.1). Vystrihnuté časti génu sa nazývajú *intróny* (*introns*) a ponechané časti *exóny* (*exons*). Ďalšia komplikácia je spôsobená tým, že preklad z mRNA do proteínu nevyužíva úplne celú mRNA. Úsek pred prvým prekladaným kodónom sa zvykne označovať 5'UTR a úsek od posledného kodónu do konca mRNA sa nazýva 3'UTR, kde UTR je skratka z anglického výrazu *untranslated region* (*neprekladaná oblasť*).

Keď teda namapujeme už spracovanú mRNA späť na miesta v genóme, odkiaľ sa transkribovala, dostaneme striedajúce sa úseky exónov a intrónov a exóny môžeme ďalej rozdeliť na kódujúce a neprekladané oblasti (obr. 4.1). Na to, aby sme vedeli podľa genetického kódu získať sekvenciu proteínu kódovaného daným génom, potrebujeme poznať presné hranice všetkých kódujúcich oblastí tohto génu. Úlohou hľadania génov je nájsť kódujúce oblasti všetkých génov v danej DNA sekvencii. Pre jednoduchosť v tejto kapitole nebudeme uvažovať hľadanie neprekladaných oblastí génov, hoci aj toto je dôležitý problém, najmä pri štúdiu regulácie transkripcie. Po tomto zjednodušení si môžeme predstaviť proces hľadania génov ako ofarbovanie jednotlivých báz v DNA sekvencii tromi farbami, pričom jedna farba zodpovedá kódujúcim oblastiam, jedna intrónom medzi kódujúcimi oblasťami v rámci génu a tretia zodpovedá medzigénovým oblastiam oddeľujúcim gény, pričom do týchto oblastí budeme zahŕňať po našom zjednodušení aj neprekladané úseky a intróny v nich (obr. 4.2). Vo väčšine textu budeme predpokladať, že každá báza má práve jednu z týchto troch úloh, t.j. je buď časťou kódujúcej oblasti, intrónu, alebo medzigénovej oblasti. Ako si ale neskôr ukážeme, tento predpoklad nie je v genómoch vždy splnený.

Už teda vieme, čo je našim cieľom: určiť pre každú bázu jednu z troch úloh resp. farieb. Nie je však zatiaľ jasné, na základe akého kritéria je možné tieto tri typy oblastí od seba odlíšiť. Jednou z vlastností, ktoré budeme využívať, sú charakteristické sekvenčné motívy na hraniciach kódujúcich oblastí. Prvý kodón v géne je napríklad vždy trojica ATG. Tieto tri bázy teda budú tvoriť začiatok prvej kódujúcej oblasti v géne. Posledný kodón génu je jeden zo stop kodónov TAA, TAG a TGA, ktoré nekódujú žiadnu aminokyselinu. Podobne prvé dve bázy každého intrónu sú takmer vždy GT a posledné dve AG (pozri



Obr. 4.2: Cieľom hľadania génov je nájsť presné hranice všetkých kódujúcich exónov všetkých génov v danej DNA sekvencii, čo si môžeme predstaviť ako ofarbenie tejto sekvencie tromi farbami. Na tomto obrázku je hypotetický gén s tromi kódujúcimi exónmi a dvoma intrónmi. Veľkými písmenami je zvýraznený štart kodón a stop kodón a tiež dve bázy typické pre začiatok a koniec intrónu.



Obr. 4.3: Príklad niekoľkých sekvencií na hranici exónu a intrónu z ľudského genómu. Prvé dve bázy intrónu sú takmer vždy GT.

veľké písmená na obr. 4.2). Niekoľko báz okolo takéhoto veľmi silne zachovaného jadra má tiež určité špecifické preferencie. V ukážke na obrázku 4.3 majú napríklad skoro všetky exóny na poslednom mieste bázu G. V bunke existujú RNA molekuly alebo proteíny, ktoré sa preferenčne viažu na miesta s určitým sekvenčným motívom a tým pomáhajú nájsť správne miesta v géne pri zostrihu a inicializácii transkripcie. Informácia obsiahnutá v sekvenčných motívoch nám však nestačí na jednoznačné nájdenie génov, lebo takéto krátke skupiny báz sa môžu vyskytovať aj inde, než na hraniciach kódujúcich exónov.

Druhým typom informácie, ktorý nám pomáha rozlíšiť kódujúce a nekódujúce oblasti genómu, sú líšiace sa frekvencie výskytov rôznych skupín báz. V tabuľke 4.1 vidíme, že frekvencie jednotlivých báz sa mierne líšia medzi exónmi, intrónmi a medzigénovými oblasťami, to by však samo o sebe nestačilo na detekciu génov. Avšak keď spočítame frekvenciu výskytu dlhších skupín báz, napríklad šestíc, dostaneme výraznejšie rozdiely medzi kódujúcimi a nekódujúcimi oblasťami. Vyplýva to z povahy kódujúcich oblastí, ktoré majú vďaka kodónom trojperiodickú štruktúru. Napríklad stop kodóny TAA, TGA a TAG sa vyskytujú len na konci posledného kódujúceho exónu. Jednotlivé amino kyseliny sú rôzne často využívané v proteínoch, čo má vplyv na frekvenciu výskytu kodónov, ktoré ich kódujú. Aj medzi kodónmi, ktoré kódujú tú istú amino kyselinu, majú často niektoré

Tabuľka 4.1: Frekvencia výskytu báz A, C, G, T v jednotlivých typoch oblastí v ľudskom genóme. Kódujúce oblasti sú rozdelené na prvé, druhé a tretie pozície v rámci kodónu.

	A	C	G	T
Kódujúci exón, pozícia 1	0.26	0.26	0.32	0.16
Kódujúci exón, pozícia 2	0.30	0.24	0.20	0.26
Kódujúci exón, pozícia 3	0.17	0.32	0.31	0.20
Intrón	0.26	0.22	0.22	0.30
Medzig. oblasť	0.27	0.23	0.23	0.27

vyššiu frekvenciu ako ostatné. Pre určitú oblasť DNA sekvencie sa teda môžeme snažiť vyhodnotiť, či sa jej zloženie podobá skôr na typické kódujúce alebo typické nekódujúce sekvencie. Informácia, ktorú takto dostaneme však opäť nie je jednoznačná.

4.1.1 Hľadanie génov ako bioinformatický problém

Ako sme videli, gény a ich kódujúce exóny majú určité charakteristické prvky, ako sú motívy na hraniciach exónov a odlišné frekvencie skupín báz. Avšak žiaden z týchto prvkov nám nedáva jednoduché pravidlo, ktoré by nám umožnilo správne rozpoznať všetky gény.

Aby sme teda problém hľadania génov definovali informaticky, použijeme opäť skórovací systém, ktorý priradí určité skóre každej možnej anotácii, teda každému rozloženiu génov a ich exónov. Cieľom je potom nájsť pre danú sekvenciu anotáciu s najvyšším skóre. Skórovací systém by mal dať vysoké skóre anotáciam, v ktorých sa jednotlivé exóny začínajú a končia motívmi, ktoré sa podobajú na známe príklady a v ktorých úseky označené ako kódujúce majú typické štatistické vlastnosti kódujúcich úsekov. Pri hľadaní génov sa často používajú pravdepodobnostné modely, ktoré umožňujú zvoliť takéto skórovanie systematickým spôsobom.

Pravdepodobnostný model na hľadanie génov si môžeme predstaviť ako hypotetické zariadenie, ktoré dokáže vygenerovať náhodnú sekvenciu S aj s anotáciou (ofarbením) A. Ak toto zariadenie budeme púšťať znova a znova, bude generovať nové a nové sekvencie a anotácie. Pri veľkom počte opakovaní sa vygenerované sekvencie a anotácie začnú opakovať, niektoré dvojice sa však budú objavovať častejšie a iné menej často. Môžeme sa teda pýtať, aká je pravdepodobnosť, že pri nasledujúcom použití zariadenia dostaneme práve sekvenciu S a anotáciu A, túto pravdepodobnosť označíme $\Pr(S, A)$. Cieľom je zostrojiť zariadenie tak, aby páry S, A s vlastnosťami podobnými skutočným génom mali veľkú pravdepodobnosť. Páry, ktoré úplne odporujú našim znalostiam o génoch, môžu mať naopak nulovú pravdepodobnosť, čo znamená, že ich zariadenie nikdy nevygeneruje.

Takéto zariadenie sa môže zdať zbytočné, lebo pri hľadaní génov nechceme generovať nové náhodné sekvencie. Naopak, máme už danú sekvenciu S a zaujíma nás jej anotácia. Táto sekvencia S sa môže na výstupe nášho zariadenia objaviť spárovaná s veľmi veľa rôznymi anotáciami, avšak s niektorými sa objavuje častejšie a s inými menej často. Výsledkom hľadania génov bude tá anotácia A, s ktorou sa sekvencia S objavuje najčastejšie, teda $A = \arg\max_A \Pr(S, A)$ (obr. 4.4). Pravdepodobnostný model nám teda definuje skórovací systém: skóre anotácie A bude jednoducho $\Pr(S, A)$.

Stále sme však nevyriešili základný problém, ako presne priradiť skóre, alebo v našom prípade pravdepodobnosť, každej dvojici S a A. Úplná tabuľka všetkých možností, ako

aa	0.008	ac	0.009	ag	0.008	…	tt	0.009
aa	0	…		…			…	
aa	0.011							
aa	0.010							
aa	0.009							
aa	0							
aa	0.007							
aa	0.001							
aa	0.010							

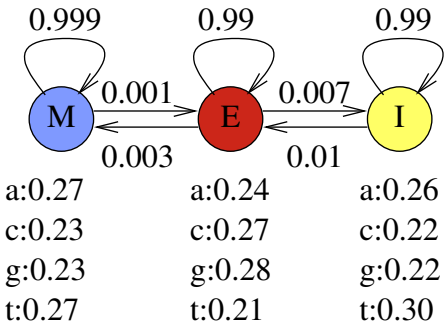
Obr. 4.4: Hračkársky príklad pravdepodobnostného modelu pre sekvencie dĺžky dva. Tabuľka má 16 stĺpcov predstavujúcich všetky dvojice nukleotidov od `aa` po `tt`. Riadky tabuľky predstavujú všetky rôzne anotácie sekvencie dĺžky dva. Pre každú oanotovanú sekvenciu tabuľka reprezentuje pravdepodobnosť, že práve ju by model vypísať pri nasledujúcom generovaní. Súčet čísel v celej tabuľke by mal byť 1. Ak máme danú sekvencou dĺžky dva, oanotujeme ju anotáciou v najvyšším číslom v danom stĺpci tabuľky. Napríklad v sekvencii $S =$`aa` ofarbíme prvú bázu modrou a druhú červenou, lebo číslo 0.011 je najvyššie v stĺpci prislúchajúcom `aa`.

na obrázku 4.4, sa dá použiť iba pre extrémne krátke sekvencie a v takých samozrejme nehľadáme gény, lebo vo veľmi krátkej sekvencii nie je možné rozlíšiť, či ide o kódujúcu oblasť, nekódujúcu oblasť alebo prechod medzi nimi. Namiesto toho použijeme skryté Markovove modely, ktoré nám umožňujú v kompaktnom tvare zapísať pravdepodobnostné rozdelenie pre ľubovoľne dlhé sekvencie.

4.2 Skryté Markovove modely

Skryté Markovove modely (*hidden Markov models*, *HMMs*) a ich rôzne rozšírenia sa používajú na viacero bioinformatických problémov, ale aj v iných oblastiach informatiky. Ako sme popísali v predchádzajúcej časti, HMM si môžeme predstaviť ako zariadenie, ktoré generuje náhodnú sekvenciu spolu s jej anotáciou (ofarbením). HMM pozostáva z niekoľkých stavov, ktoré pri hľadaní génov zodpovedajú rôznym typom oblastí, ako napríklad exóny, intróny a medzigénové oblasti. Na obrázku 4.5 vidíme jednoduchý HMM na hľadanie génov s troma stavmi. Stavy sú poprepájané prechodmi, ktoré majú určené pravdepodobnosti. Pri generovaní náhodnej sekvencie sa presúvame medzi stavmi, pričom v každom kroku si vyberieme ďalší stav podľa pravdepodobností prechodov, ktoré z neho vychádzajú. Napríklad ak sme v modeli na obr. 4.5 v stave M, tak s pravdepodobnosťou 0.999 zostaneme v tom istom stave, s pravdepodobnosťou 0.001 prejdeme do stavu E.

V každom kroku tiež vygenerujeme jednu bázu DNA sekvencie, pričom každý stav má svoju vlastnú tabuľku emisných pravdepodobností, ktorá určuje pravdepodobnosti vygenerovania jednotlivých báz. Napríklad v stave M je pravdepodobnosť vygenerovania bázy A 27%, kým v stave E je pravdepodobnosť iba 24%. V procese generovania teda začne v určitom stave. V každom kroku vygenerujeme jednu bázu podľa aktuálneho stavu a presunieme sa do ďalšieho stavu, ktorý môže byť aj ten istý, ako aktuálny. Po n krokoch tohto procesu vygenerujeme n báz, pričom každá báza je oanotovaná farbou stavu, ktorý ju vygeneroval. Pravdepodobnosť, že za n krokov vygenerujeme konkrétnu sekvenciu S a konkrétnu anotáciu A je súčinom pravdepodobností prechodu a emisie, ktoré v priebehu generovania boli použité. Napríklad ak určíme, že model na obrázku 4.5 vždy začína v stave M, dostávame nasledovné pravdepodobnosti pre sekvenciu `aca` a dve konkrétne



Obr. 4.5: Skrytý Markovov model s troma stavmi. Stav M zodpovedá medzigénovým úsekom, stave E kódujúcim oblastiam a stav I intrónom. Pravdepodobnosti prechodu sú uvedené pri jednotlivých šípkach, emisné pravdepodobnosti sú v tabuľke pod každým stavom.

a	E	I	M
E	0.99	0.007	0.003
I	0.01	0.99	0
M	0.001	0	0.999

e	a	c	g	t
E	0.24	0.27	0.28	0.21
I	0.26	0.22	0.22	0.30
M	0.27	0.23	0.23	0.27

π	E	I	M
	0	0	1

Obr. 4.6: Tabuľky pravdepodobností a, e a π pre model na obrázku 4.5.

anotácie:

$$\Pr(S = \texttt{aca}, A = MEE) = 0.27 \cdot 0.001 \cdot 0.27 \cdot 0.99 \cdot 0.24 = 0.000017$$

$$\Pr(S = \texttt{aca}, A = MMM) = 0.27 \cdot 0.999 \cdot 0.23 \cdot 0.999 \cdot 0.27 = 0.017$$

Vidíme, že anotácia MMM má oveľa vyššiu pravdepodobnosť ako MEE a ak by sme si rozpísali aj pravdepodobnosť všetkých ostatných anotácií sekvencie `aca`, vyšlo by nám, že anotácia MMM má najvyššiu pravdepodobnosť, je to teda anotácia, ktorú by sme na základe tohoto modelu vybrali. Podobne by sme ale vedeli vyrátať v tomto modeli aj pravdepodobnosť pre oveľa dlhšie sekvencie a anotácie.

4.2.1 Formálna definícia HMM

Po tomto neformálnom úvode si zaveďme matematické označenie pre prácu s HMM. Množinu stavov modelu označíme Q a množinu emitovaných symbolov Σ (v našom prípade $\Sigma = \{\texttt{a}, \texttt{c}, \texttt{g}, \texttt{t}\}$). Model je určený troma tabuľkami pravdepodobností a, e a π. Tabuľka prechodových pravdepodobností a určuje pre každé dva stavy $u, v \in Q$ pravdepodobnosť prechodu $a(u, v)$ zo stavu u do stavu v. Ak prechod z u do v nie je možný, táto pravdepodobnosť je 0. Tabuľka emisných pravdepodobností e určuje pre každý stav $u \in Q$ a každý symbol $x \in \Sigma$ pravdepodobnosť $e(u, x)$ vygenerovania symbolu x v stave u. Napokon, tabuľka počiatočných pravdepodobností π určuje pre každý stav $u \in Q$ počatočnú pravdepodobnosť $\pi(u)$, teda pravdepodobnosť, že proces generovania začne v tomto stave. Na obrázku 4.6 vidíme tabuľky pravdepodobností pre HMM z obrázku 4.5.

Tabuľky musia určovať rozdelenie pravdepodobnosti, teda obsahujú nezáporné čísla a pre každý stav musí byť súčet emisných pravdepodobností 1 a súčet prechodových pravdepodobností pre vychádzajúce prechody musí byť tiež 1. Formálne teda pre každé

$u \in Q$ platí $\sum_{x \in \Sigma} e(u,x) = 1$ a $\sum_{v \in Q} a(u,v) = 1$. Aj počiatočné pravdepodobnosti musia mať súčet 1: $\sum_{v \in Q} \pi(v) = 1$.

Zaujíma nás pravdepodobnosť, že model určený takou sadou tabuliek v n krokoch vygeneruje sekvenciu $S = S_1, \ldots, S_n$ a anotáciu $A = A_1, \ldots, A_n$. Táto pravdepodobnosť je definovaná ako súčin hodnôt z tabuliek:

$$\Pr(A_1, \ldots, A_n, S_1, \ldots, S_n) = \pi(A_1) e(A_1, S_1) \prod_{i=2}^{n} a(A_{i-1}, A_i) e(A_i, S_i)$$

Teraz sme teda úplne definovali HMM: je to sada tabuliek, ktorá spĺňa podmienky uvedené vyššie a ktorá určuje pravdepodobnosť pre každú sekvenciu S a anotáciu A.

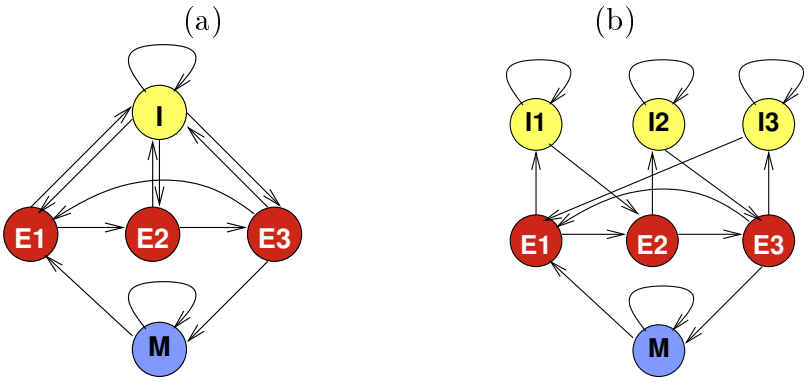
4.2.2 Použitie HMM na hľadanie génov

Ukázali sme si, ako s použitím tabuliek určujúcich HMM spočítať pravdepodobnosť $\Pr(S, A)$ pre sekvenciu S a anotáciu A. Aby sme mohli použiť HMM na hľadanie génov, musíme najskôr zvoliť tieto tabuľky. Tvorba modelu na hľadanie génov sa väčšinou robí v dvoch krokoch. V prvom kroku si ručne zvolíme množinu stavov a určíme povolené prechody medzi nimi. Tu využijeme naše znalosti biologického problému, ktorý sa snažíme riešiť. Napríklad v modeli na obrázku 4.5 sme vytvorili jeden stav pre každý typ oblasti, ktoré chceme rozlišovať a zakázali sme prechody medzi medzigénovou oblasťou a intrónom, lebo predpokladáme, že každý gén začína aj končí exónom. Neskôr si ukážeme aj zložitejšie modely na hľadanie génov s väčším počtom stavov.

Na určenie pravdepodobností v tabuľkách použijeme trénovaciu množinu anotovaných sekvencií, t.j. sekvencií, v ktorých už boli dostatočne spoľahlivo nájdené gény. V tejto trénovacej množine spočítame frekvencie zodpovedajúce hodnotám v tabuľkách modelu. Napríklad nájdeme všetky bázy v exónoch a pre každú sa pozrieme, aké je anotácia bázy, ktoré za ňou nasleduje. Spočítame, koľko percent exónových báz je nasledovaných ďalšou exónovou bázou, koľko percent je nasledovaných intrónom a koľko percent medzgénovou oblasťou. Takto spočítané frekvencie uložíme do prvého riadku tabuľky a na obrázku 4.6. Podobne na tvorbu emisnej tabuľky vezmeme napríklad všetky bázy anotované ako kódujúci exón a spočítame, koľko percent z nich je A, C, G a T, tieto čísla uložíme do prvého riadku tabuľky e na obrázku 4.6. Ak máme takto natrénované tabuľky modelu, model bude generovať sekvencie, ktoré sa na skutočné trénovacie sekvencie budú podobať v základných štatistických vlastnostiach, ako napríklad priemerný počet exónov v géne, ich priemerná dĺžka a frekvencie výskytu jednotlivých báz v rôznych oblastiach.

Pri trénovaní modelu treba dbať na to, aby namerané frekvencie boli dostatočne reprezentatívne pre genóm ako celok. Trénovacia množina teda musí byť dosť veľká. Navyše, rôzne genómy majú rôzne štatistické vlastnosti. Preto je najlepšie použiť trénovaciu množinu z genómu, ktorý chceme anotovať, alebo z blízko príbuzného organizmu.

Keď potom hotový model chceme použiť na hľadanie génov, potrebujeme pre danú sekvenciu S nájsť anotáciu A s najvyššou pravdepodobnosťou $\arg\max_A \Pr(S, A)$. Pre každú anotáciu A vieme $\Pr(S, A)$ ľahko spočítať, ale možných anotácií je veľmi veľa, takže nie je možné ich všetky skontrolovať a nájsť tú najlepšiu. Našťastie na hľadanie najlepšej anotácie existuje efektívny Viterbiho algoritmus založený na dynamickom programovaní. Jeho časová zložitosť je $O(nm^2)$, kde n je dĺžka sekvencie a m je počet stavov modelu.



Obr. 4.7: (a) Rozšírené HMM pre hľadanie génov, v ktorom je pre každú pozíciu v kodóne jeden stav. (b) Rozšírené HMM pre hľadanie génov s troma kópiami stavu pre intrón, ktoré zabezpečujú správnu nadväznosť pozícií v kodóne pred intrónom a za intrónom.

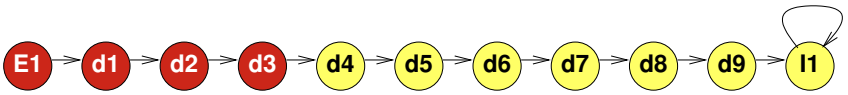
4.2.3 Tvorba modelu pre hľadanie génov

Model na obrázku 4.5 neobsahuje dostatok informácie na to, aby sa ním dali hľadať gény s dostatočnou presnosťou. Nevieme v ňom vyjadriť mnohé známe vlastnosti génov, ktoré sme opisovali v časti 4.1. V tejto časti si ukážeme niekoľko rozšírení, ktorými sa dostaneme bližšie k modelom, ktoré sa v praxi používajú na hľadanie génov.

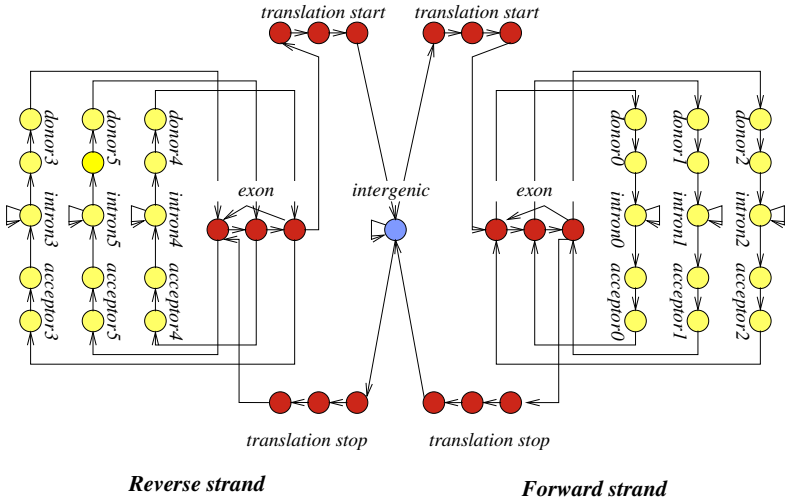
Ako vieme, kódujúce sekvencie pozostávajú z kodónov, trojíc kódujúcich jednotlivé amino kyseliny. Model teda môžeme rozšíriť tak, aby mal tri stavy pre kódujúce oblasti, každý zodpovedajúci jednej pozícii v rámci kodónu (obr. 4.7a). Tieto tri stavy sú spojené v cykle tak, aby po skončení jedného prechodu kodónom sa model vrátil zase späť do prvého stavu. Stavy pre jednotlivé pozície majú rôzne emisné pravdepodobnosti, napríklad podľa tabuľky 4.1. Z medzigénovej oblasti vstupujeme vždy do stavu pre prvú pozíciu kodónu, lebo gén začína vždy celým kodónom. Naopak zo stavu pre poslednú pozíciu kodónu je možné prejsť späť do stavu pre medzigénovú oblasť. Do intrónu je možné prejsť z hociktorého exónového stavu, lebo intrón sa môže nachádzať aj uprostred kodónu.

Problém predchádzajúceho modelu je v tom, že po návrate z intrónového stavu sa nemusí vrátiť do správnej pozície v rámci kodónu a že teda celý kódujúci úsek nemusí mať dĺžku deliteľnú tromi. Tento problém je vyriešený v modeli na obrázku 4.7b vytvorením troch kópií stavu pre intrón. Ak napríklad exón končí po prvej báze kodónu, model prejde do prvej kópie intrónu. Z nej je možné prejsť do druhého stavu pre exón, čím sa zabezpečí, že model po prerušení správne pokračuje v kódujucej oblasti. Ak nemáme vedomosti o tom, že by sa intróny výraznejšie štatisticky líšili podľa polohy v rámci kodónu, môžeme všetkým trom intrónovým stavom priradiť tie isté emisné pravdepodobnosti.

Nášmu modelu zatiaľ chýba možnosť reprezentovať sekvenčné motívy na hraniciach exónov, ako napríklad miesto zostrihu z obrázku 4.3. Takýto motív môžeme do modelu pridať ako reťaz stavov, ktorou nahradíme priamy prechod zo stavu pre exón do stavu pre intrón (obr. 4.8). Každý zo stavov reprezentuje jeden znak motívu a jeho emisné pravdepodobnosti zodpovedajú frekvenciám báz na danom mieste motívu. Napríklad prvé dve bázy v intróne sú takmer vždy GT. Stav zodpovedajúci prvej báze v intróne (stav d_4 na obr. 4.8) teda bude mať pravdepodobnosť 1 pre emitovanie G a pravdepodobnosť 0 pre emitovanie ostatných báz.



Obr. 4.8: Reťaz stavov reprezentujúca sekvenčný motív na konci exónu. Stav E_1 reprezentuje stred kódujúceho exónu (prvú pozíciu v kodóne), stavy d_1, d_2, d_3 reprezentujú posledné tri bázy exónu so špecifickým motívom, stavy $d_4, \ldots d_9$ reprezentujú prvých šesť báz intrónu a stav I_1 reprezentuje stred intrónu.



Obr. 4.9: HMM na hľadanie génov s génmi na oboch vláknach a s krátkymi signálmi na hraniciach exónov.

Na obrázku 4.9 vidíme model, ktorý má krátke motívy na všetkých hraniciach exónov a tiež správne zachováva kodóny aj keď sú prerušené intrónom. V tomto modeli máme ešte jednu kópie stavov pre gény, ktorá reprezentuje gény na opačnom vlákne. V genómoch sú totiž gény premiešané na oboch vláknach chromozómov. Keď model používame na sekvenciu referenčného vlákna, gén na opačnom vlákne začína stop kodónom, samozrejme v komplementárnych bázach, takže napríklad stop kodón TAA na opačnom vlákne bude na referenčnom vlákne ako TTA. Aj celý zvyšok génu nasleduje v opačnom poradí a v komplementárnych bázach. Preto v tejto časti modelu otočíme smer prechodov a v tabuľke emisných pravdepodobností zmeníme bázy za komplementárne.

Aj po všetkých týchto rozšíreniach ešte model nie je použiteľný na hľadanie génov, lebo v stavoch pre kódujúce a nekódujúce úseky máme v tabuľkách uložené len frekvencie jednotlivých báz a nie ich skupín. Nevieme napríklad zabezpečiť ani to, aby sa uprostred kódujúceho úseku nevyskytoval stop kodón. Tento problém sa dá najjednoduchšie vyriešiť použitím stavov vyšších rádov. Bežné HMM, ktorými sme sa zaoberali doteraz, majú stavy nultého rádu. Tabuľky emisných pravdepodobností určujú pravdepodobnosť vygenerovania určitej bázy v závislosti od aktuálneho stavu. Stav rádu k má väčšiu tabuľku, v ktorej pravdepodobnosť aktuálnej bázy závisí od aktuálneho stavu a od k predchádzajúcich báz. Ak by sme napríklad použili pre tretiu pozíciu kodónu rád aspoň 2, vedeli by sme sa vyhnúť stop kodónom, lebo ak by napríklad predchádzajúce dve bázy boli TA, bázy A a G budú mať pravdepodobnosť 0, lebo TAA aj TAG sú stop kodóny. Ak by ale predchádzajúce dve bázy boli GA, mohli by sme použiť hociktorú bázu s nenulovou pravdepodobnosťou.

Stav k-teho rádu teda určuje frekvencie skupín dĺžky $k+1$ v sekvencii a má v svojej emisnej tabuľke 4^{k+1} pravdepodobností (ak HMM emituje DNA sekvencie). Nakoľko veľkosť tabuliek rastie exponenciálne rýchlo v závislosti od rádu, musíme zvoliť pomerne malú hodnotu k aby tabuľky neboli príliš veľké a aby sme mali dosť trénovacích dát na ich určenie. V praxi sa na charakterizáciu kódujúcich a nekódujúcich oblastí používajú stavy rádu okolo päť, čo nám umožňuje zachytiť frekvencie dvojíc kodónov v kódujúcich oblastiach. V sekvenčných motívoch tiež existujú závislosti medzi susdnými, alebo aj nesusednými pozíciami, väčšinou však máme dosť dát len na modely pomerne nízkeho rádu, medzi 0 a 2.

Programy na hľadanie génov používané v praxi často nepoužívajú základne HMM popísané v tejto kapitole, ale rôzne ich zovšeobecnenia. Jedným takým rozšírením je možnosť v stave vygenerovať viac ako jednu bázu, pričom počet vygenerovaných báz sa tiež náhodne vygeneruje z určitého pravdepodobnostného rozdelenia. To nám umožňuje modelovať typické rozdelenie dĺžok exónov a iných oblastí. Nevýhodou je spomalenie Viterbiho algoritmu.

4.3 Hľadanie génov v praxi

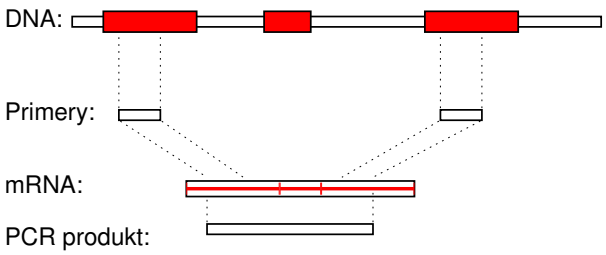
Skryté Markovove modely, ktoré sme videli v predchádzajúcej časti, tvoria základ viacerých používaných programov na hľadanie génov. Avšak ich anotácie nie sú vždy úplne správne. V tejto časti sa budeme zaoberať limitáciami týchto programov, stratégiami na zlepšovanie ich presnosti a inými praktickými aspektami hľadania génov.

4.3.1 Overovanie génov a prídavná informácia

Ako sme videli, na trénovanie modelov na hľadanie génov potrebujeme vytvoriť trénovaciu množinu známych génov. Takisto výsledky hľadačov génov nie sú vždy správne, takže potrebujeme ich správnosť overiť iným spôsobom. Existuje niekoľko postupov na experimentálne určovanie alebo overovanie génov.

Pomerne priamočiare je overovania transkripcie a zostrihu. Zo vzorky buniek je možné vyextrahovať mRNA (s už vystrihnutými intrónmi) a tieto sekvencie potom sekvenovať. V minulosti sa používalo Sangerovo sekvenovanie, pričom z každej mRNA sa získal jeden segment (read) dlhý niekoľko stoviek báz. Takéto sekvencie sa nazývajú EST (expressed sequence tag). V súčasnosti sa používajú sekvenovacie technológie novej generácie, ktorými vieme v jednom behu sekvenovacieho prístroja získať veľké množstvo krátkych sekvencií zo vzorky mRNA. Tento postup sa nazýva RNA-Seq (pozri aj kap. 1). Získané EST alebo RNA-Seq dáta môžeme namapovať na genóm technikami na zarovnávanie sekvencií. Sekvenovaná mRNA je veľmi podobná na genomické sekvenciu z ktorej bola transkribovaná, môžu sa však vyskytovať drobné rozdiely kvôli sekvenovacím chybám alebo polymorfizmom. Získaná skevencia mRNA však môže tiež pochádzať z viacerých exónov, takže pri jej mapovaní získame viacero lokálnych zarovnaní, ktoré susedia v mRNA ale sú oddelené intrónmi v DNA. Oblasti, na ktoré sa mapujú získané mRNA sekvencie, predstavujú exóny a oblasti oddeľujúce lokálne zarovnania tej istej mRNA sekvencie predstavujú intróny.

Sekvenovaním mRNA teda vieme nájsť v genóme oblasti, ktoré sú transkribované. Táto technológia však má tiež určité problémy. Sekvenovaná mRNA môže byť konta-



Obr. 4.10: Technógia RT PCR na overovanie transkripcie génov.

minovaná buď genomickou DNA alebo pre-mRNA obsahujúcou intróny. Teda aj oblasti, ktoré v skutočnosti neboli transkribované, sa môžu nachádzať v EST knižniciach. Naopak niektoré gény sú transkribované len v malých množstvách alebo len za určitých špecifických podmienok a teda nemusia byť vôbec zastúpené vo vzorke, ktorú sme sekvenovali. Problém môže vzniknúť aj pri mapovaní sekvencií na genóm, lebo niektoré sekvencie sa môžu dať namapovať na viacero miest v genóme a nevieme, z ktorej oblasti skutočne pochádzajú.

Sekvenovanie mRNA je vhodné na celogenómové skúmanie transkripcie, niekedy však potrebujeme overiť transkripciu a zostrih jedného konkrétneho génu. Na to slúži technológia RT-PCR (reverse transcription polymerase chain reaction). Máme teda predpovedaný gén a z jeho exónov si zvolíme dva krátke kúsky sekvencie nazývané primery (obr. 4.10). Tieto primery sa syntetizujú a spolu s ďalšími potrebnými chemikáliami sa pridajú k vzorke mRNA. Ak vo vzorke existuje transkript obsahujúci obidva primery vo vhodnej vzdialenosti, RT-PCR vytvorí veľké množstvo kópií oblasti transkriptu medzi primermi. Túto oblasť potom môžeme sekvenovať a zistiť presné miesta zostrihu. Ak sa však v žiadnom transkripte tieto dva primery nenachádajú, nedostaneme v RT-PCR žiaden produkt. Aj táto technológia nám však dokáže nájsť iba gény prítomné v danej vzorke a môže dôjsť ku kontaminácii alebo problémom s mapovaním. Je však pomerne citlivá a dokáže zachytiť aj transkripty prítomné iba vo veľmi malých množstvách.

Technológie založené na sekvenovaní RNA však nedokážu rozlíšiť medzi génmi kódujúcimi proteínmi a inými transkribovanými oblasťami, napríklad RNA génmi. Ideálne by sme teda chceli overiť aj transláciu, teda prítomnosť výsledného proteínu v bunke. Overovanie prítomnosti proteínov je náročnejší proces ako overovanie RNA. Na celogenómovej úrovni sa používajú technológie založené na hmotnostnej spektrometrii (mass spectrometry), ktoré merajú hmotnosti krátkych fragmentov proteínov a porovnávajú ich s vypočítanými hmotnosťami fragmentov predpovedaných génov. Daný predpovedaný proteín je možné aj syntetizovať v bakteriálnych bunkách, pripraviť k nemu špecifickú protilátku a touto protilátkou testovať na jeho prítomnosť vo vzorkách. Tento proces je však veľmi časovo aj finančne náročný.

Nepriame svedectvo o správnosti predpovedaných génov poskytuje aj komparatívna genomika, teda porovnávanie genómov rôznych organizmov. Najjednoduchší spôsob je porovnávať predpovedané gény s databázou známych proteínov. Ak sa predpovedaný proteín podobá na nejaký známy proteín, je šanca, že ide o skutočný gén s podobnou funkciou. Môže však tiež ísť iba o náhodnú podobnosť alebo o pseudogén, pozostatok kedysi funkčného génu. Ak máme niekoľko blízko príbuzných genómov, môžeme zostaviť ich viacnásobné celogenómové zarovnanie a snažiť sa nájsť evolučné stopy génov aj bez

znalosti anotácie v jednom z nich. Oblasti kódujúce proteíny totiž zvyčajne majú viac synonymných mutácií, ktoré nemenia proteín, než nesynonymných, o čom si povieme viac v kapitole 5.

Všetky tieto zdroje dát o polohe génov môžeme použiť viacerými spôsobmi. Jedna možnosť je použiť ich v počiatočných fázach anotácie genómu na vytvorenie ručne overenej sady spoľahlivých génov podporených experimentálnymi dátami. Takéto gény môžeme použiť na trénovanie hľadača génov a v neskorších fázach aj na testovanie, teda meranie presnosti automatickej anotácie. Je však potrebné oddeliť gény používané na trénovanie od tých, ktoré neskôr použijeme na testovanie, lebo na trénovacích génoch modely môžu dosahovať lepšie výsledky než na génoch, ktoré neboli pri trénovaní použité a teda náš odhad presnosti by bol príliš optimistický. Druhá možnosť je použiť externé zdroje dát priamo v programe na hľadanie génov na zvýšenie jeho presnosti. Napríklad ak sa určitej oblasti genómu namapujú segmenty z RNA-Seq experimentu, môžeme predpokladať, že ide o exón a môžeme nejakým spôsobom zvýšiť pravdepodobnosť anotácií v HMM, ktoré na tomto mieste exón majú.

4.3.2 Príklady programov na hľadanie génov

Na hľadanie génov existuje veľké množstvo programov. V tejto kapitole sa zaoberáme hlavne hľadaním génov v eukaryotických genómoch. V prokaryotických genómoch je problém hľadania génov jednoduchší, nakoľko väčšinou nemajú intróny. Programy ako napríklad GeneMark Lukashin and Borodovsky (1998) a Glimmer Delcher et al. (1999) sú pomerne jednoducho použiteľné a niektoré obsahujú aj trénovací modul, ktorý prispôsobí parametre modelu skúmanému genómu.

V eukarotických genómoch medzi prvé HMM na hľadanie génov patril HMMGene Krogh (1997), ktorého autor patrí medzi priekopníkov využitia HMM v bioinformatike. Po dlhé roky bol štandardom program Genscan Burge and Karlin (1997) založený na zovšeobecnených HMM, ktoré umožňujú modelovanie dĺžok exónov. Podobné modely využívali aj novšie programy, napríklad GeneZilla Majoros et al. (2004), ExonHunter Brejová et al. (2005) a Augustus Stanke and Waack (2003). Najnovšia generácia programov (napríklad CONTRAST Gross et al. (2007) a CONRAD DeCaprio et al. (2007)) je založená na type modelov nazývaných conditional random fields a dosahuje presnejšie výsledky. Vo všeobecnosti je však trénovanie eukaryotických hľadačov génov pomerne náročné a nie vždy úplne automatizované.

Mnohé programy tiež používajú na zvýšenie presnosti celogenómové zarovnania viacerých genómov. Twinscan Korf et al. (2001) bol prvý úspešný program používajúci dva genómy. Neskôr Exoniphy Siepel and Haussler (2004) vedel využiť aj viacero genómov, nehľadá ale celé gény iba exóny a N-SCAN Gross and Brent (2006) je úspešným rozšírením Twinscanu na viacero genómov.

Okrem celogenómových zarovnaní programy môžu využívať aj iné zdroje dát, napríklad zarovnania známych proteínov alebo sekvenovaných transkriptov ku genómu (napríklad programy ExonHunter Brejová et al. (2005), Augustus Stanke et al. (2006), Jigsaw Allen and Salzberg (2005), Fgenesh++ Solovyev et al. (2006) a ďalšie).

Obr. 4.11: Príklady alternatívneho zostrihu. Pod sebou sú vždy znázornené dva transkripty z tej istej oblasti genómu, exóny sú hrubšie červené obdĺžniky, intróny sú vyfarbené žltou farbou. V génoch s väčším počtom exónov sa môžu vyskytovať aj rôzne kombinácie týchto foriem.

4.3.3 Obmedzenia programov na hľadanie génov

V našej definícii hľadania génov sme spravili niekoľko zjednodušení, ktoré nie vždy platia v skutočných genomickéch sekvenciách. Prvé bolo, že gény sa navzájom nepretínajú. V eukaryotických genómoch sa len zriedka prekrývajú kódujúce úseky rôznych génov, môže však napríklad nastať situácia, že v intróne jedného génu je celý kratší gén umiestnený na opačnom vlákne DNA. Mnohé program ne sú schopné takéto prekrývajúce sa gény nájsť.

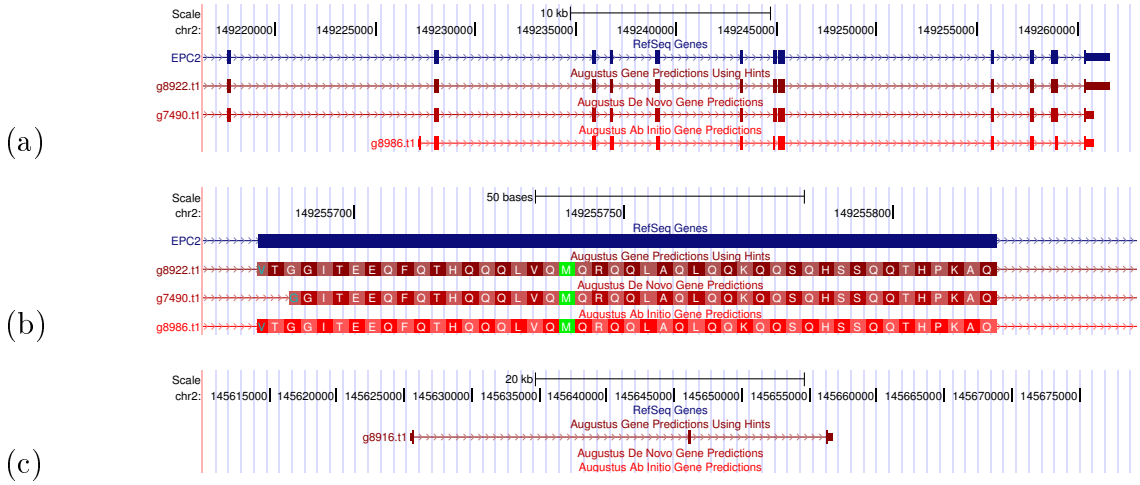
Ešte rozšírenejšou komplikáciou je alternatívny zostrih (alternative splicing). V jednom géne totiž zostrih za rôznach podmienok môže prebehnúť rôznymi spôsobmi a oblasť, ktoré je v jednom transkripte exónom môže byť v inom transkripte z toho istého génu vystrihnuté ako intrón (obr. 4.11). Ak pri hľadaní génu označíme každú bázu jednoznačne ako exón, intrón alebo medzigénový úsek, nájdeme iba jednu z možných foriem transkriptu.

Spomínali sme tiež, že programy na hľadanie génov často ignorujú neprekladané oblasti exónov a hľadajú iba oblasti kódujúce proteín. Takéto anotácie sú postačujúce, ak chceme získať prehľad o počte génov genóme a o proteínoch, ktoré kódujú. Ak však chceme podrobnejšie skúmať reguláciu transkripcie, je dobré vedieť, kde transkripcia začína a končí, lebo v okolí týchto bodov, môžu byť rôzne regulačné motívy.

Napokon netreba zabúdať na to, že programy na hľadanie génov sú založené na charakterizácii typických čŕt génov. Gény, ktoré sú svojou sekvenciu netypické, môžu teda týmto programom robiť problémy. Príkladom sú gény, ktoré používajú neobvzklé sekvenčné motívy, alebo ktoré majú veľmi krátke alebo naopak veľmi dlhé exóny či intróny.

Tieto a ďalšie vplyvy spôsobujú, že hľadače génov nevedia nájsť všetky gény úplne presne. Na obrázku 4.12 vidíme príklady výsledkov programu Augustus. Aj keď celkovo pomerne dobre zodpovedajú uznávaným anotáciam z databázy RefSeq, nájdu sa tam aj viaceré rozdiely.

V roku 2005 sa uskutočnilo vyhodnotenie viacerých programov na hľadanie génov na ľudskom genóme Guigo et al. (2006). Ich výskedky boli porovnávané s manuálne kontrolovanou anotáciou a štatisticky vyhodnotené. Najlepšie programy, ktoré okrem anotovanej DNA nepoužívali žiadnu prídavnú informáciu, našli správne okolo 60% exónov, pričom exón považujeme za nájdený správne iba ak program našiel aj správnu polohu obidvoch jeho okrajov. A iba 20% génov bolo nájdených správne, pričom gén považujeme za správne nájdený, ak program predpovedal aspoň jeden z jeho prípadných alternatívnych transkriptov úplne presne od jedného konca po druhý, vrátane všetkých exónov. Mohlo by sa zdať, že ak programy nevedia správne nájsť 80% všetkých génov, ich výsledky sú nepoužiteľné, ako sme však videli na obrázku 4.12, aj gény, ktoré nie sú äplne správne sa môžu z väčšej časti zhodovať a predpovedaná sekvencia proteínu sa nemusí príliš líšiť od tej



Obr. 4.12: Príklady anotácií z programu Augustus v ľudskom genóme zobrazené pomocou UCSC prehliadača genómov. Modrou je zobrazený gén z uznávanej databázy RefSeq, pod ním odtieňami červenej niekoľko verzií anotácií programu Augustus. Prvá z nich využívala informáciu zo zarovnaní známych proteínov a sekvenovaných transkriptov, druhá využívala porovnávanie rôznych genómov a tretia iba samotnú DNA sekvenciu človeka. Exóny sú zobrazené ako obdĺžniky, intróny ako vodorovné čiary. V časti (a) vidíme 3' koniec génu s niekoľkými exónmi. Augustus pomerne dobre zodpovedá anotácii z databázy, bez použitai prídavnej informácie však vynechal prvú časť génu. V časti (b) vidíme podrobnejší pohľad na jeden z exónov, kde vidíme, že jedna z verzií Augustusu zle určil jeho ľavý koniec. V časti (c) vidíme gén, ktorý predpovedala jedna z verzií Augustusu, ale nie je v databáze známych génov.

skutočnej.

Navyše presnosť výsledkov sa zlepší, ak použijeme ďalšie dostupné informácie. Pri použití celogenómových zarovnaní dostávame správne okolo 65% exónov a 35% génov a pri použití známych proteínov a mRNA sekvencií dostávame správne až 85% exónov a 70% génov. Ďalšie zlepšenie bolo dosiahnuté od roku 2005 použitím nových typov modelov.

Genóm človeka (a podobne aj genómy iných cicavcov) sú navyše na hľadanie génov pomerne náročné, lebo iba okolo 1% sekvencie kóduje proteíny a gén má v priemere až 10 exónov. Hľadáme teda krátke ostrovčeky kódujúcich exónov oddelené dlhými intrónami a medzigénovými úsekmi. Mnohé iné skupiny organizmov majú kompaktnejšie genómy s kratšími intrónmi, v ktorých sa oveľa ľahšie hľadajú gény.

4.3.4 Koľko génov má človek?

V tejto kapitole sme videli výpočtové metódy na hľadanie génov aj rôzne technológie na ich experimentálne overovanie. Spojením týchto prístupov ľudia už roky snažia nájsť všetky gény v ľudskom genóme a určiť ich počet.

Pred sekvenovaním ľudského genómu odhady počtu génov museli používať rôzne nepriame metódy alebo extrapolovať z malého množstva známych sekvencií. V tomto období vedci odhadovlai pomerne vysoký počet génov v rozmedzí 50 000–140 000. Keď bola v roku 2001 publikovaná predbežná verzia ľudského genómu, jej anotátori dospeli k odhadu 30 000–40 000 génov. V roku 2004 bola publikované vylepšená verzia ľudského genómu, v ktorej boli opravené alebo doplnené mnohé oblasti a bolo tiež dostupných viac experimen-

tálnych dát a lepšie programy na hľadanie génov. V tejto verzii bolo teda predpovedaných 20 000–25 000 génov. Takéto nízke číslo mnohých ľudí prekvapilo, lebo podobný počet génov má napríklad aj jednoduchý červ *Caenorhabditis elegans*. Odvtedy sa odhad počtu ľudských génov kódujúcich proteíny príliš nezmenil. V roku 2007 Clamp et al. (2007) porovnali známe katalógy génov Ensembl, RefSeq a VEGA a našli v nich spolu 24 500 rôznych génov. Z nich ale iba 20 500 malo typické znaky génov kódujúcich proteíny, takže zvyšné môžu byť buď RNA gény alebo nepsrávne predpovede. Táto práca však neposkytla žiaden odhad počtu génov, ktoré ešte nepoznáme. Aj v roku 2010 sa ešte odborníci na hľadanie génov domnievajú, že skutočný počet génov sa môže líšiť až o 1000 od počtu 22 333, čo bol aktuálny počet génov v databáze RefSeq Pertea and Salzberg (2010). Navyše genómy rôznych jedincov sa navzájom môžu líšiť v desiatkach génov, lebo mnohé gény sa vyskytujú v premenivom počte kópií. Nie je teda možné určiť jeden presný počet, ktorý by platil pre celé ľudstvo.

4.4 Zhrnutie

V tejto kapitole sme sa zaoberali hľadaním génov kódujúcich proteíny v osekvenovaných genómoch. Z takto nájdených génov môžeme potom predpovedať ich proteínové sekvencie a tie ďalej skúmať. Ide teda o jednu zo základných analýz, ktoré sa robí na novoosekvenovaných genómoch. Ukázali sme si, že gény je možné hľadať pomocou skrytých Markovových modelov, čo sú pravdepodobnostné modely v ktorých vyjadríme štatistické vlastnosti typických génov, ako napríklad frekvencie výskytu rôznych kombinácií báz v kódujúcich a nekódujúcich úsekoch a sekvenčné motívy na hraniciach exónov. Presnosť výsledkov môžeme testovať alebo aj vylepšovať využitím dát získaných z rôznych experimentov.

Kapitola 5

Komparatívna genomika

Komparatívna genomika sa venuje porovnávaniu viacerých genómov. Vďaka takémuto porovnávaniu môžeme často získať informáciu, ktorú by bolo veľmi ťažké získať zo sekvencií jednotlivých druhov. Základným nástrojom komparatívnej genomiky je využívanie faktu, že rôzne časti genómu sa počas evolúcie menia rôznymi rýchlosťami.

Rýchlosť evolučných zmien je totiž ovplyvnená *prirodzeným výberom*. Mutácie, ktoré by negatívnym spôsobom ovplyvnili schopnosť konkrétneho organizmu prežiť a rozmnožovať sa (takzvané *škodlivé mutácie; deleterious mutations*) pozorujeme v evolúcii menej často ako mutácie, ktorých vplyv na fungovanie organizmu je neutrálny (*neutrálne mutácie*). Miesta, kde škodlivé mutácie môžu nastať, sú ovplyvnené tzv. *purifikačným výberom* (purifying selection). Prejavy purifikačného výberu môžeme pozorovať všade tam, kde sa nachádzajú v genóme funkčné elementy, napr. gény kódujúce proteíny, časti genómu zodpovedné za reguláciu, či časti kódujúce dôležité molekuly RNA. Porovnávanie viacerých sekvencií nám tak môže pomôcť pri hľadaní takýchto funkčných prvkov v genóme.

Existujú však aj mutácie, ktoré môžu ovplyvniť organizmus pozitívnym spôsobom (takzvané *prospešné mutácie; advantageous mutations*). Sú to napríklad mutácie, vďaka ktorým gén nadobudne nový typ funkcie alebo ktoré prispejú k optimalizácii už existujúcej funkcie génu. Miesta, kde takéto mutácie nastanú, sú ovplyvnené *pozitívnym výberom* (positive selection). Nájdenie miest ovplyvnených pozitívnym výberom nám umožňuje identifikovať tie mutácie, ktoré môžu byť priamo zodpovedné za odlišnosti medzi organizmami. Napríklad genómy človeka a šimpanza sa od seba odlišujú zhruba v jednom percente pozícií, čo predstavuje zhruba 20 miliónov pozícií, kde sa nukleotid v genóme človeka odlišuje od zodpovedajúceho nukleotidu v genóme šimpanza. Len malá časť z týchto nukleotidov však zodpovedá skutočným funkčným zmenám, ktoré sú ale zodpovedné napríklad za obrovské rozdiely vo fungovaní mozgu, schopnosti používať reč a ďalšie schopnosti špecifické pre ľudí.

Metódy v oblasti komparatívnej genomiky často používajú nasledujúci spoločný postup. Vychádza sa z celogenómových zarovnaní tak, ako sme ich popísali v kapitole 2.7. Zvolíme si jeden *referenčný genóm*, ku ktorému zarovnáme všetky ostatné genómy, s ktorými ho chceme porovnávať. Celogenómové zarovnania v tomto prípade použijeme na to, aby sme lokalizovali úseky z ostatných genómov, ktoré pochádzajú z tej istej sekvencie najbližšieho spoločného predka (*ortológy*).

Výsledkom takéhoto predspracovania sú viacnásobné zarovnanie ortologických sekvencií, ktoré môžu vyzerať ako zarovnanie na obrázku 5.1. Takéto zarovnanie je zložené

```
Človek AGTGGCTGCCAGGCTG---GGATGCTGAGGCCTTGTTTGCAGGGAGGT
 Makak AGTGGCTGCCAGGCTG---GGTTGCTGAGGCCTTGTTTGCCGGGAGGT
   Myš GGTGGCTGCCGGGCTG---GGTGGCTGAGGCCTTGTTGGTGGGGTGGT
   Pes AGTGGCTGCCCGGCTG---GGTGGCTGAGGCCTTATTTGCAGGGAGGT
   Kôň GATGGCTGCCGGGCTG---GGCTGCCGAGGCCTTGTTCGTGGGGAGGT
Pásovec AGTGGCTGCCGGGCTG---GGAGGCCAAGGCCTTGTTCGCGGGCAGGT
Sliepka AGTGGCTGCCAGTCTGCGCCGTGGCCGACGTCTTGCTCGGGGGAAGGT
   Žaba AATGGCTTCCATTTTGTGCCGCTGCTGAGGTCTTGTTCTGGGGAAGAT
```

Obr. 5.1: Príklad viacnásobného zarovnania, ktoré možno použiť na analýzu pomocou metód komparatívnej genomiky.

z množstva stĺpcov, každý stĺpec reprezentuje evolúciu danej bázy v jednotlivých organizmoch.

Jedným z možných prístupov ku komparatívnej genomike je tvorba pravdepodobnostných modelov, ktoré jednak modelujú evolúciu jednotlivých báz (stĺpcov) zarovnania pomocou modelov evolúcie tak, ako sme si ich predstavili v kapitole 3 a spájajú túto informáciu s modelmi ako sú napríklad skryté Markovove modely (viď kapitola 4, aby zachytili rozdiely medzi časťami genómu, ktoré sú pod rôznymi typmi selekčných tlakov. Bližšie si tieto prístupy predstavíme v ďalších častiach tejto kapitoly.

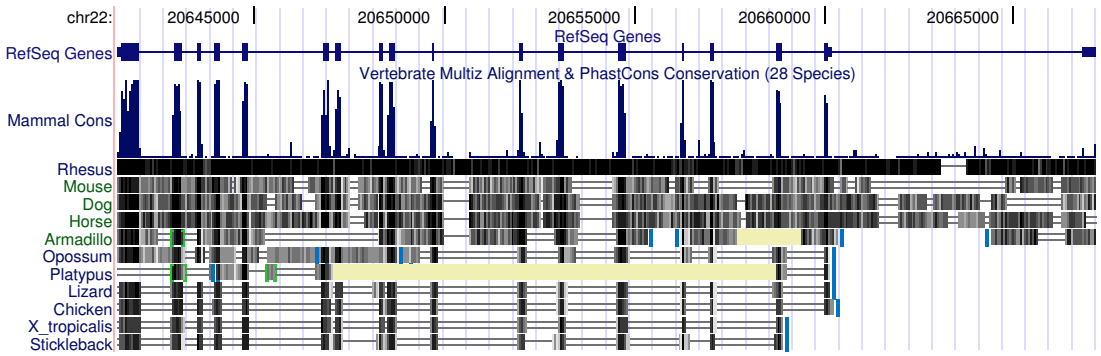
5.1 Štúdium purifikačného výberu

Vďaka purifikačnému výberu sa funkčné časti genómu vyvíjajú pomalším tempom ako nefunkčné. Preto sa funkčné časti môžeme snažiť nájsť tak, že budeme hľadať časti sekvencie, ktoré sú v evolúcii viac zachované ako ich okolie. Veľká časť takto získaných kandidátov budú zodpovedať známym typom funkčných elementov—génom kódujúcim proteíny, regulačným sekvenciám a podobne.

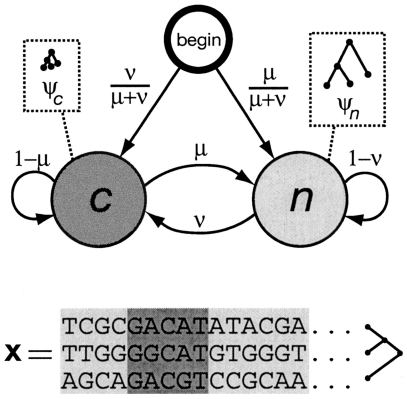
Ak napríklad vezmeme celogenómové zarovnanie genómov človeka a myši, zistíme že až 98% exónov kódujúcich proteíny je pokrytých týmto zarovnaním (Mouse Genome Sequencing Consortium, 2002) a priemerná zhoda zarovnaných sekvencií v exónoch je približne 85%. Na druhej strane, z intrónov je iba 48% pokrytých zarovnaním, u ostatných intrónov nevieme ani spoľahlivo lokalizovať ortologickú sekvenciu v myši. Navyše zarovnané sekvencie intrónov medzi človekom a myšou sa zhodujú iba na 69%. Rozdiely v divergencii funkčných a nefunkčných sekvencií sú teda v tomto prípade značné.

Na obrázku 5.2 je zobrazená anotácia niekoľkoexónového ľudského génu v UCSC genome browseri. Pod anotáciou je zobrazená úroveň identity zarovnaných sekvencií z niekoľkých organizmov, pričom tmavšia farba znamená vyššiu podobnosť a dvojitá čiara znamená úsek, ktorý nie je pokrytý zarovnaním. Riadok označený ako *mammal conservation* sumarizuje úroveň zachovania sekvencie pomocou štatistickej analýzy. Vidíme, že takáto analýza dokáže jasne odlíšiť exóny od intrónov.

Na hľadanie zachovaných sekvencií pod vplyvom purifikačného výberu môžeme použiť napríklad *fylogenetický skrytý Markovov model*. Je to pravdepodobnostný model, ktorý spája modely evolúcie (viď kapitola 3), charakterizujúce mutácie v jednotlivých stĺpcoch zarovnania, spolu so skrytým Markovovým modelom, ktorý charakterizuje rozloženie za-



Obr. 5.2: Ukážka z UCSC browsera zobrazujúca rozdiely úrovne zachovania sekvencie intrónov a exónov v rôznych organizmoch.



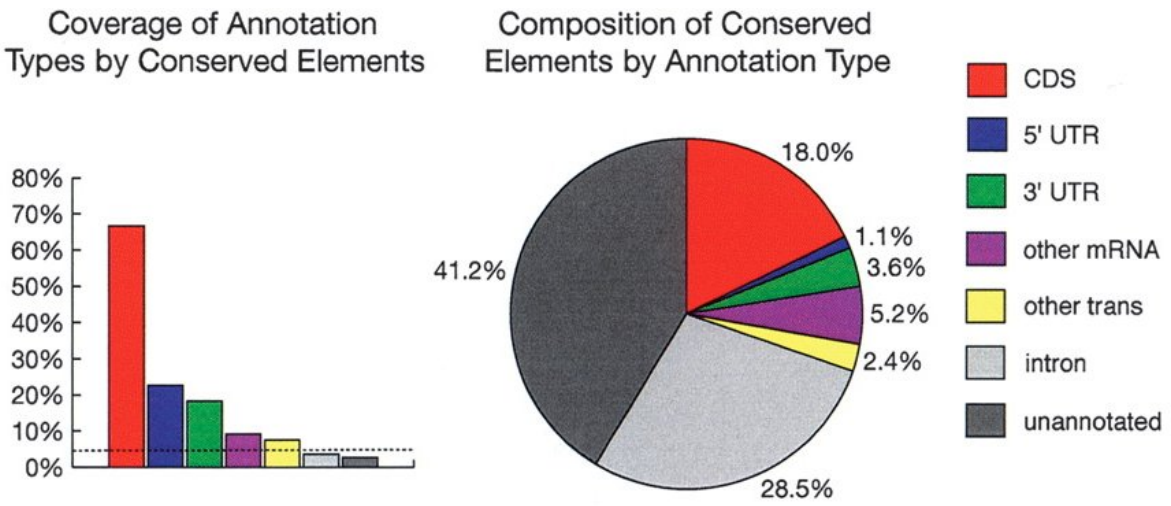
Obr. 5.3: Fylogenetický skrytý Markovov model, ktorý je podkladom pre program PhastCons. Zdroj: Siepel et al. (2005)

chovaných a nezachovaných oblastí v genóme.

Schéma takéhoto modelu je zobrazená na obrázku 5.3. Okrem počiatočného stavu má model dva stavy: stav c bude reprezentovať sekvencie pod vplyvom negatívnej selekcie (funkčné sekvencie), kým druhý stav n reprezentuje sekvencie, ktoré podliehajú neutrálnej evolúcii. Takto model repreyentuje zarovnanie DNA sekvencií, v ktorom sa striedajú úseky pod vplyvom negatívnej selekcie s neutrálnymi úsekmi, pričom rozdelenie dĺžok týchto úsekov je určené pravdepodobnosťami prechodu μ a ν.

Namiesto jednej bázy DNA však náš model bude emitovať v každom kroku celý stĺpec zarovnania. Na to využijeme pravdepodobnostný model fylogenetického stromu, ktorý sme opísali v kapitole 3.5.1. V tomto prípade predpokladáme, že usporiadanie jednotlivých organizmov v strome, ako aj dĺžky hrán sú pevne dané. Dĺžky hrán však prenásobíme v každom stave inou konštantou: v stave c konštantou ψ_c a v stave n konštantou ψ_n. Konštanty nastavíme tak, aby $\psi_c < \psi_n$. Toto spôsobí, že stav c bude generovať stĺpce zarovnania, kde sekvencie v listoch sú si navzájom podstatne podobnejšie, ako sekvencie generované v stave n, keďže hrany fylogenetického stromu asociovaného so stavom c sú kratšie ako hrany v prípade stavu n. Výsledný model potom generuje zarovnania sekvencií spolu s označením, či ide o zachované časti (na obrázku tmavé) alebo neutrálne časti (na obrázku svetlé).

Tak ako v prípade hľadania génov, takýto model určuje pravdepodobnostné rozdele-



Obr. 5.4: Výsledky aplikácie programu PhastCons na celogenómové zarovnania genómov človeka, myši, sliepky a fugu. (vľavo) Parametre modelu boli nastavené tak, aby cca 5% genómu bolo pokrytých výslednými zachovanými úsekmi (čiarkovaná čiara). Obrázok ukazuje aká časť špecificky oanotovaných častí genómu je pokrytá predpovedanými úsekmi. (vpravo) Rozdelenie predpovedaných zachovaných úsekov podľa ich oanotovanej funkcie. Jednými z najzaujímavejších sú predpovedané úseky, ktoré doposiaľ nemajú žiadnu známu funkciu. Zdroj: Siepel et al. (2005)

nie nad všetkými možnými zarovnaniami a ich anotáciami. Anotáciou je v tomto prípade rozdelenie zarovnania na zachované a neutrálne časti. Na to, aby sme takýto model použili, nám teraz stačí pre danú sekvenciu nájsť najpravdepodobnejšiu anotáciu. Toto je možné urobiť pomocou inferenčného algoritmu, ktorý je kombináciou Viterbiho algoritmu pre inferenciu v skrytých Markovových modeloch (kapitola 4) a Felsensteinovho algoritmu (kapitola 3.5.1), ktorý nám umožňuje spočítať vierohodnosť rôznych fylogenetických stromov zodpovedajúcich jednotlivým stavom modelu.

Jednou z prvých aplikácií takéhoto modelu na úrovni celého genómu je program PhastCons (Siepel et al., 2005). Obrázok 5.4 ukazuje zhrnutie výsledkov aplikácie tohto programu na celogenómové zarovnanie ľudského genómu s genómami myši, sliepky a ryby fugu. Obrázok demonštruje dva dôležité fakty. Za prvé, časti genómu so známou funkciou (napr. gény kódujúce proteíny, ako aj ďalšie oanotované funkčné elementy) sú obohatené o evolučne zachované sekvencie, kým časti genómu, pre ktoré žiadna funkcia nie je anotovaná, obsahujú menej zachovaných sekvencií, ako by sa očakávalo pri ich náhodnom rozmiestnení v genóme (obrázok vľavo). Takýto záver jasne poukazuje na to, že skutočne existuje prepojenie medzi zachovanými sekvenciami predpovedanými pomocou pravdepodobnostného modelu a funkciou jednotlivých častí genómu. Za druhé, zhruba 40% evolučne zachovaných sekvencií nemá v súčasnosti priradenú žiadnu známu funkciu (obrázok vpravo). Takouto analýzou teda získavame značné množstvo nových kandidátov, úsekov DNA, ktoré môžu mať dôležitú funkciu pre život organizmu, no táto funkcia doposiaľ nie je známa.

5.2 Hľadanie génov pomocou komparatívnej genomiky

Ako sme videli v predchádzajúcej časti, veľké množstvo evolučne zachovaných častí genómu tvoria práve sekvencie kódujúce proteíny (CDS). Preto je logické, že takúto informáciu by sme chceli využiť aj na spresnenie predikcie samotných génov.

V zarovnaniach genómov je však obsiahnuté podstatne väčšie množstvo informácie užitočnej pri hľadaní génov ako len to, či sú sekvencie viac alebo menej zachované. Kľúčom k efektívnemu využitiu takejto informácie je genetický kód (tabuľka 5.1), ktorý určuje, ako sa trojice nukleotidov (kodóny) v génoch prekladajú do aminokyselín, ktoré potom formujú proteíny.

Zoberme si napríklad kodón CTC, ktorý kóduje aminokyselinu leucín. Mutáciou C→A na tretej pozícii získame kodón CTA, ktorý opäť kóduje leucín, z hľadiska kódovaného proteínu teda nenastala žiadna zmena. Mutácie, ktoré nemenia kódovanú aminokyselinu, nazývame *synonymné*, tieto mutácie by nemali mať veľký vplyv na funkciu výsledných proteínov. Na druhej strane, mutácia T→A na druhej pozícii (vznikne kodón CAC) spôsobí zmenu kódovanej aminokyseliny na histidín. Takáto mutácia mení kódovaný proteín a potenciálne ovplyvňuje aj jeho funkciu. Mutácie, ktoré menia kódovanú aminokyselinu nazývame *nesynonymné*. Kým synonymné mutácie sú z hľadiska výberu väčšinou neutrálne, nesynonymné mutácie sú často pod vplyvom silnej negatívnej selekcie.

Toto je možné aj veľmi jednoducho demonštrovať vďaka štruktúre genetického kódu. Mutácie na prvých dvoch pozíciách kodónu sú totiž často nesynonymné, kým mutácie na tretej pozícii sú takmer vždy synonymné. Preto ak porovnáme DNA sekvencie kódujúce proteíny v rôznych organizmoch, mali by sme na tretej pozícii vidieť podstatne nižšiu mieru zachovania, než na prvých dvoch pozíciách. Skutočne v takýchto zarovnaniach medzi človekom a myšou vidíme na prvej pozícii 82% zachovaných báz, na druhej pozícii až 87%, na tretej pozícii je však iba 61% zachovaných báz. Mutácie v úsekoch, ktoré kódujú proteíny, budú mať teda špecifickú trojperiodickú štrukúru, kde prvé dve pozície kodónu sú obvykle veľmi dobre zachované, kým tretia pozícia vykazuje podstatne vyššiu variabilitu.

Tento fakt ale už vieme veľmi ľahko využiť pri hľadaní génov. Základná myšlienka je použiť opäť fylogenetické skryté Markovove modely, pričom stavy a prechody použijeme podobné ako u HMM na hľadanie génov, ale pre každý stav použijeme evolučný model s inou dĺžkou hrán: pre prvé dve pozície kodónu budú hrany stromu kratšie (a teda modelované sekvencie budú zachovanejšie), kým pre tretie pozície a intróny použijeme strom s dlhšími hranami. V rôznych obmenách túto myšlienku používa viacero hľadačov génov (viď napr. Exoniphy (Siepel and Haussler, 2004) a N-SCAN (Gross and Brent, 2006)). Použitie zarovnaní môže signifikantne zvýšiť presnosť hľadania génov, ako ukazuje tabuľka 5.2.

5.3 Štúdium pozitívneho výberu

Doteraz sme sa zaoberali tým, že sme využívali existenciu purifikačného výberu u funkčných sekvencií na to, aby sme takéto funkčné sekvencie hľadali v zarovnaniach viacerých druhov a odlíšili ich od neutrálne vyvíjajúcich sa sekvencií. Purifikačný výber zaisťuje zachovanie ich funkcie a jeho prejavom je "spomalenie" rýchlosti evolúcie vo funkčných sekvenciách. Pozitívny výber naopak vidíme u sekvencií, ktoré získavajú nové funkcie,

Alanine (A)
GC*

Cysteine (C)
TGC
TGT

Aspartic acid (D)
GAC
GAT

Glutamic acid (E)
GAA
GAG

Phenylalanine (F)
TTC
TTT

Glycine (G)
GG*

Histidine (H)
CAC
CAT

Isoleucine (I)
ATA
ATC
ATT

Lysine (K)
AAA
AAG

Leucine (L)
CT*
TTA
TTG

Methionine (M)
ATG

Asparagine (N)
AAC
AAT

Proline (P)
CC*

Glutamine (Q)
CAA
CAG

Arginine (R)
CG*
AGA
AGG

Serine (S)
TC*
AGT
AGC

Threonine (T)
AC*

Valine (V)
GT*

Tryptophan (W)
TGG

Tyrosine (Y)
TAC
TAT

Stop codon (*)
TAA
TAG
TGA

Tabuľka 5.1: Genetický kód určuje, akým spôsobom sa trojice nukleotidov (kodóny) prekladajú do aminokyselín.

	Exóny		Gény	
Program	sn	sp	sn	sp
AUGUSTUS (1 genóm)	52%	63%	24%	17%
NSCAN (zarovnanie)	68%	82%	35%	37%

Tabuľka 5.2: Porovnanie hľadačov génov AUGUSTUS (1 genóm) a N-SCAN (viac genómov) podľa Guigo et al. (2006).

prípadne sa ich funkcia výrazným spôsobom mení. Jedným zo základných prejavov pôsobenia pozitívneho výber u sekvencií kódujúcich gény je prítomnosť veľkého množstva nesynonymných mutácií. Tento fakt sa budeme snažiť využiť na ich hľadanie.

Na prvý pohľad by sa mohlo zdať, že stačí vziať zarovnanie sekvencií konkrétneho génu u niekoľkých druhov, spočítať počet synonymných a nesynonymných mutácií, ktoré v takomto zarovnaní vidíme a z týchto čísel vyvodiť závery. Kapitoly 3.4-3.5 nám však už ukázali, že to celé nie je až také jednoduché: počas evolúcie na jednom mieste v DNA sekvencii môže totiž dôjsť k viacerým mutáciám, v konečnej sekvencii však vidíme už iba finálny výsledok. Preto nevieme jednoznačne spočítať počet mutácií, ktoré počas evolúcie skutočne nastali a tobôž nevieme rozhodnúť, koľko z týchto mutácií bolo synonymných a koľko nesynonymných. Aby sme tento problém vyriešili, použijeme opäť pravdepodobnostné modelovanie.

5.3.1 Zovšeobecnené modely evolúcie

V kapitole 3.4 sme si ukázali jednoduchý Jukes-Cantorov model evolúcie nukleotidov, v ktorom sa mutácie z ľubovoľnej bázy na ľubovoľnú inú diali s rovnakou pravdepodobnosťou. Pre prax je takýto model príliš jednoduchý, preto sa používajú rôzne modely, kde sa rýchlosti mutácií medzi jednotlivými bázami líšia. Vo všeobecnosti takýto model možno vyjadriť nasledujúcou 4×4 *maticou rýchlostí (rate matrix)*:

$$\mu = \begin{pmatrix} \cdot & \mu_{AC} & \mu_{AG} & \mu_{AT} \\ \mu_{CA} & \cdot & \mu_{CG} & \mu_{CT} \\ \mu_{GA} & \mu_{GC} & \cdot & \mu_{GT} \\ \mu_{TA} & \mu_{TC} & \mu_{TG} & \cdot \end{pmatrix} \tag{5.1}$$

(Hodnoty na diagonále matice sa dopočítavajú podľa ostatných hodnôt a v tomto okamihu nie sú pre nás podstatné.) Hodnota μ_{xy} v tejto matici vyjadruje rýchlosť, akou sa určitá báza x mení na inú bázu y. Formálnejšie, ak $\Pr(X_{t+\Delta} = y \,|\, X_t = x)$ je pravdepodobnosť, že za čas Δ sa zmení báza x na bázu y, potom

$$\mu_{xy} = \lim_{\Delta \to 0} \frac{\Pr(X_{t+\Delta} = y \,|\, X_t = x)}{\Delta}. \tag{5.2}$$

Ak maticu rýchlostí celú prenásobíme nejakou konštantou α, jednoducho zrýchlime celý model evolúcie α krát, takže vlastne iba zmeníme časovú jednotku, s ktorou v celom modeli pracujeme. Preto nás viac ako samotné čísla v matici budú zaujímať pomery jednotlivých hodnôt.

Jedným z najpoužívanejších modelov pri analýze evolúcie DNA sekvencií je *HKY model* nazvaný podľa autorov Hasegawa, Kishino a Yano (Hasegawa et al., 1985). Tento model oproti Jukes-Cantorovmu modelu obsahuje dve inovácie. Za prvé, rozlišuje dve triedy báz, ktoré sa značne líšia svojou biochemickou štruktúrou: pyrimidíny (C, T) a puríny (A, G). Mutácie v rámci triedy (*tranzície*) sa vďaka podobnosti štruktúr dejú častejšie ako mutácie medzi triedami (*transverzie*). Pomer medzi tranzíciami a transverziami označíme ako parameter HKY modelu κ.

Ak nastavíme čas v evolučnom modeli na nekonečno, nezáleží na tom, z ktorej bázy sme začali. Frekvencia výskytu jednotlivých báz sa ustáli v tzv. *ekvilibriu*. V Jukes-Cantorovom modeli je pravdepodobnosť ľubovoľnej bázy v ekvilibriu 1/4. Pri aplikácii

Jukes-Cantorovho modelu by sa teda zloženie DNA sekvencií približovalo k rovnakému zastúpeniu všetkých báz, čo nezodpovedá realite. S tým súvisí druhá zmena HKY modelu: jednotlivé rýchlosti mutácií sú nastavené tak, aby zodpovedali reálnym frekvenciám nukleotidov v analyzovaných sekvenciách. Frekvencie jednotlivých nukleotidov označíme π_A, π_C, π_G a π_T (samozrejme, $\pi_A + \pi_C + \pi_G + \pi_T = 1$) a budú takisto parametrami modelu.

Pre parametre $\kappa, \pi_A, \pi_C, \pi_G, \pi_T$ HKY model používa nasledujúcu maticu rýchlostí:

$$\mu_{x,y} = \begin{cases} \kappa\pi_y & \text{ak mutácia } x \to y \text{ je tranzícia,} \\ \pi_y & \text{ak mutácia } x \to y \text{ je transverzia.} \end{cases} \tag{5.3}$$

Z tejto matice je možné podobne ako pri Jukes-Cantorovom modely pre ľubovoľné x, y a Δ vypočítať pravdepodobnosť $\Pr(X_{t+\Delta} = y \,|\, X_t = x)$. Frekvencie jednotlivých nukleotidov v ekvilibriu sú $\pi_A, \pi_C, \pi_G, \pi_T$.

5.3.2 Model evolúcie kodónov

Pri analýze pozitívneho výberu nebudeme uvažovať model evolúcie jednotlivých nukleotidov, ale model, ktorý uvažuje celé kodóny (trojice nukleotidov) naraz. Na tejto úrovni totiž vieme rozlíšiť u jednotlivých substitúcií, či sú synonymné alebo nesynonymné.

Model evolúcie kodónov zostavíme rovnakým spôsobom, ako model evolúcie nukleotidových sekvencií. Bude opäť charakterizovaný maticou rýchlostí, ovšem táto matica nebude rozmerov 4×4, ale 61×61, keďže máme 64 možných rôznych trojíc nukleotidov, z ktorých 3 zodpovedajú stop kodónom, takže zostáva 61 kodónov kódujúcich amino kyseliny.

Kodónový model má nový parameter ω, ktorý reprezentuje *pomer rýchlostí nesynonymných mutácií k synonymným mutáciám*. Ak teda zvolíme $\omega < 1$, pôjde o model sekvencie pod vplyvom purifikačného výberu, $\omega = 1$ zodpovedá neutrálne sa vyvíjajúcim sekvenciám a pre $\omega > 1$ dostaneme model evolúcie kodónov pod vplyvom pozitívneho výberu.

Nech π_x je frekvencia výskytu kodónu x v skutočných sekvenciách a nech κ je pomer tranzícií a transverzií. Potom náš model evolúcie kodónov bude určený nasledujúcou maticou (x a y sú v tomto prípade kodóny, teda trojice nukleotidov):

$$\mu_{x,y}(\omega) = \begin{cases} 0 & \text{ak medzi } x \text{ a } y \text{ je viac ako jeden rozdiel,} \\ \pi_y & \text{ak sa } x \text{ a } y \text{ líšia v synonymnej transverzii,} \\ \kappa\pi_y & \text{ak sa } x \text{ a } y \text{ líšia v synonymnej tranzícii,} \\ \omega\pi_y & \text{ak sa } x \text{ a } y \text{ líšia v nesynonmnej transverzii,} \\ \omega\kappa\pi_y & \text{ak sa } x \text{ a } y \text{ líšia v nesynonymnej tranzícii.} \end{cases} \tag{5.4}$$

Z takéhoto modelu možno rovnakým spôsobom ako v prípade nukleotidových modelov pre ľubovoľné kodóny x a y a pre ľubovoľný čas Δ určiť pravdepodobnosť, že po uplynutí času Δ kodón x zmutuje na kodón y. Všimnime si, že model neumožňuje priamu zmenu kodónu x na kodón y v prípade, že sa x od y líšia vo viac ako jednom nukleotide. To však neznamená, že by pre takéto kodóny bola pravdepodobnosť $\Pr(X_{t+\Delta} = y \,|\, X_t = x, \omega)$ tiež nulová. Zmenu z x na y totiž možno zrealizovať pomocou viacerých substitúcií, s čím výpočet pravdepodobnosti počíta. Tak ako v prípade HKY modelu, frekvencie kodónov v ekvilibriu budú zodpovedať parametrom π_x.

5.3.3 Fylogenetický strom a vierohodnosť parametru ω

Teraz keď máme evolučný model, ktorý zahŕňa parameter ω, môžeme tento model použiť pre pevne daný fylogenetický strom a pre danú hodnotu ω na výpočet pravdepodobnosti vygenerovanie dát v danom zarovnaní. Stačí postupovať rovnakým spôsobom ako v kapitole 3.5.1 s tým, že jeden stĺpec zarovnania je v tomto prípade stĺpec kodónov (tzn. pozeráme sa na tri bázy namiesto jednej) a namiesto Jukes-Cantorovho modelu použijeme model pre kodóny, ktorý sme navrhli vyššie.

Samozrejme, dôležitá je voľba parametru ω. Pre rôzne hodnoty ω dostaneme totiž rôzne pravdepodobnosti vygenerovania dát v našom zarovnananí. Hodnotu takejto pravdepodobnosti pre dané ω nazveme *vierohodnosť parametru* ω.

Jednou z možností, ako určiť, či ide o gén, ktorý je pod vplyvom pozitívnej selekcie, je zobrať zarovnanie kodónov tohto génu a nájsť ω s maximálnou pravdepodobnosťou vygenerovania takýchto dát, inak povedané nájsť ω, ktoré maximalizuje vierohodnosť. Takúto optimalizáciu ω je možné urobiť pomocou štandardných numerických metód. Ak výsledné ω je menšie ako 1, pôjde o sekvenciu, ktorá je pod vplyvom purifikačného výberu, ak je $\omega > 1$, ide o sekvenciu pod vplyvom pozitívnej selekcie.

Táto metóda by v praxi pomerne dobre fungovala pre veľmi dlhé gény, kde nastalo dostatok nesynonymných aj synonymných zmien, aby odhad ω pomocou maximalizácie vierohodnosti bol dostatočne spoľahlivý. Pre kratšie gény, kde je počet mutácií menší, je však táto metóda veľmi citlivá na štatistický šum. Preto je dôležité použiť ďalšiu štatistickú metódu, ktorá nám pomôže zistiť, či je výsledok dostatočne spoľahlivý. Túto metódu popíšeme v ďalšej časti kapitoly.

5.3.4 Test pomeru vierohodností (likelihood ratio test)

Skúsme sa zamyslieť nad tým, čo by sa stalo s maximálnou vierohodnosťou v prípade, že by sme obmedzili možné hodnoty ω na interval medzi 0 a 1 (teda nedovolíme, aby model vysvetlil dáta pôsobením pozitívneho výberu). Sú dve možnosti, ktoré závisia od hodnoty maximálnej vierohodnosti ω^* v modeli bez takéhoto obmedzenia:

- Ak $\omega^* \leq 1$, potom aj pri obmedzení hodnôt ostáva ω^* najvierohodnejšou hodnotou parametra, a teda sa vierohodnosť nijakým spôsobom nezmení.
- Ak $\omega^* > 1$, potom je táto hodnota mimo povoleného intervalu. Nová najvierohodnejšia hodnota bude niektorá z hodnôt ω medzi 0 a 1, a teda v takto modifikovanom modeli sa aj vierohodnosť dát zníži.

Ak teda vykonáme optimalizáciu v pôvodnom aj modifikovanom modeli, v druhom prípade dostaneme buď rovnakú vierohodnosť (ak v skutočnosti nešlo o pozitívny výber) alebo nižšiu vierohodnost (ak šlo o pozitívny výber). Samozrejme, nižšiu vierohodnosť môžeme dostať aj z dôvodu atypickosti alebo malého množstva dát (v takom prípade dostaneme pre veľké množstvo hodnôt ω podobné hodnoty vierohodnosti). Pri veľkom poklese vierohodnosti je však takáto situácia nepravdepodobná.

Aký je teda dostatočný pokles vierohodnosti na to, aby sme mohli prehlásiť, že ide o štatisticky významný rozdiel? Nech L_1 je maximálna vierohodnosť v prípade modelu s neobmedzeným ω (alternatívny model) a L_0 je maximálna vierohodnosť v prípade modelu

s obmedzením $\omega < 1$ (nulový model). V takomto prípade môžeme použiť nasledujúcu štatistiku: $D = -2 \log L_0/L_1$.

Test pomerom vierohodností (*likelihood ratio test*) nemôže odmietnuť nulovú hypotézu v prípade, že štatistika D je príliš malá. Dáta, ktoré sme videli, mohli pri takomto výsledku vzniknúť práve tak dobre v modeli bez pozitívnej selekcie, ako v modeli s pozitívnou selekciou. Naproti tomu, pri dostatočne veľkom D pravdepodobnosť toho, že ide o nulovú hypotézu (P-hodnota), je veľmi malá. Vzťah medzi hodnotou štatistiky D a P-hodnotou je určený Wilksovou vetou, ktorá hovorí, že rozdelenie hodnôt D v prípade, že skutočné dáta pochádzajú z modelu podľa nulovej hypotézy, sa asymptoticky blíži ku $\chi^2(1)$ distribúcii.

Pri teste pozitívneho výberu teda postupujeme nasledovným spôsobom:

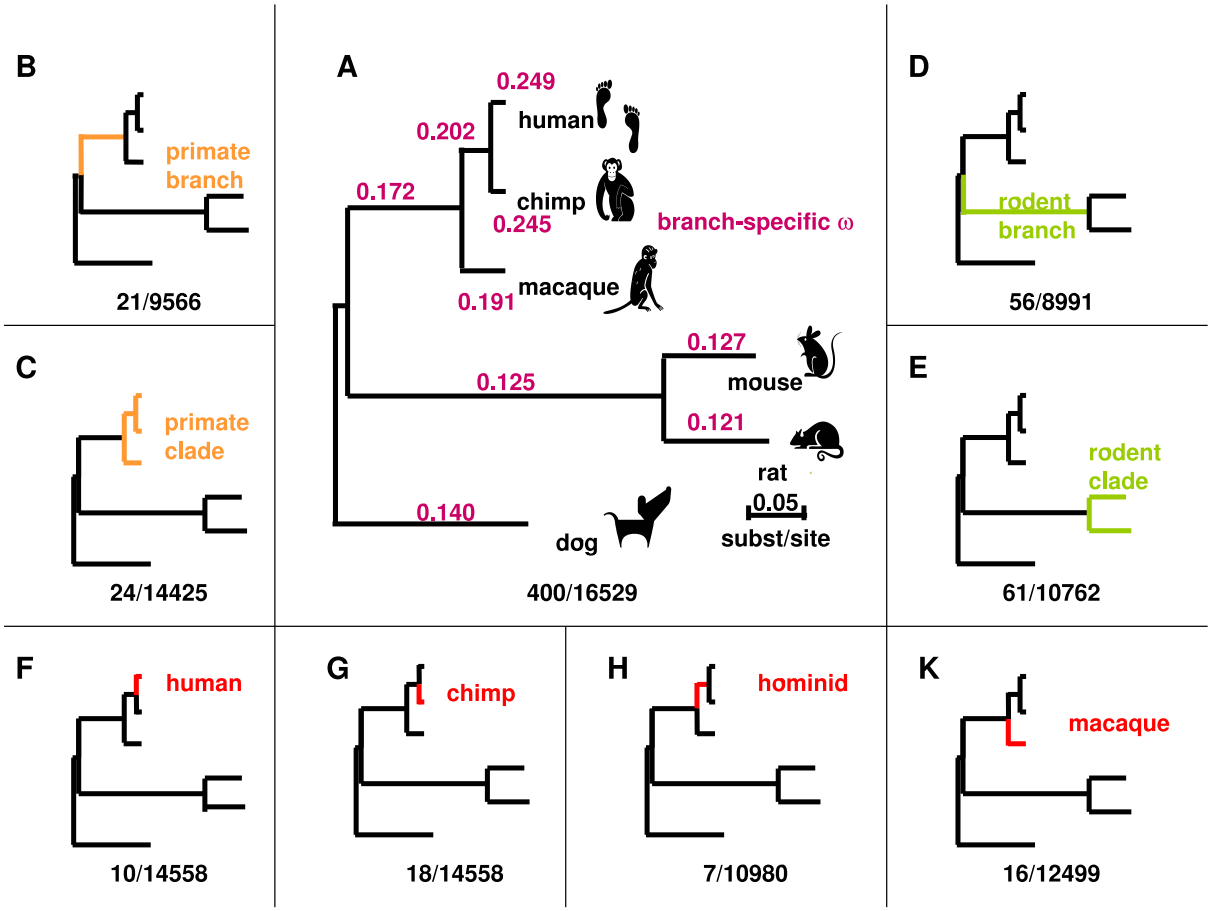
1. Pre daný gén vypočítame maximálne vierohodnosti L_0 (obmedzený model) a L_1 (neobmedzený model).
2. Vypočítame štatistiku $D = -2 \log L_0/L_1$.
3. Zistíme príslušný kvantil hodnoty D v $\chi^2(1)$ distribúcii, ktorý nám určuje P-hodnotu. (Napr. ak $D > 3.841$, potom $P < 0.05$.)
4. Ak P-hodnota je dostatočne nízka, môžeme o géne prehlásiť, že je pod vplyvom pozitívneho výberu. V opačnom prípade nevieme vylúčiť, že ide o purifikačný výber alebo neutrálnu evolúciu.

Poznámka: Pri aplikácii takejto metódy na celý genóm je ešte potrebné uplatniť štandardné korekcie chýb viacnásobného testovania, keďže vykonávame niekoľko tisíc štatistických testov.

5.3.5 Pozitívny výber v genómoch cicavcov

Test pozitívneho výberu je možné uplatniť celogenómovo. Napríklad analýza zarovnania genómov človeka, šimpanza, makáka, myši, potkana a psa (Kosiol et al., 2008) identifikovala 400 génov pod vplyvom pozitívnej selekcie počas celej evolúcie cicavcov. Testy na pozitívny výber na špecifických hranách alebo kladách stromu možno realizovať tak, že v alternatívnom modeli povolíme pozitívnu selekciu iba na vybraných hranách stromu; počty génov identifikované v takýchto hranových a kladových testoch sumarizuje obrázok 5.5. Na obrázku je tiež pre každú vetvu fylogenetického stromu zobrazená najvierohodnejšia hodnota ω pre zreťazenie zarovnaní všetkých génov v genóme. Vidíme, že "obvyklá" hodnota ω je medzi 0.1 a 0.25. Z toho vyplýva, že veľká časť génov je pod vplyvom silného purifikačného výberu, tak ako sme to na začiatku predpokladali.

Veľká časť funkčných kategórií génov s neobvykle vysokým podielom génov pod vplyvom pozitívneho výberu súvisí s imunitou, obranou a senzorickými schopnosťami. Nadreprezentácia práve týchto kategórií dáva veľmi dobrý zmysel: vo všetkých týchto prípadoch ide o nutnú adaptáciu na podnety prostredia, či už je to evolúcia a výskyt nových patogénov alebo zmena životného štýlu a zdrojov jedla, ktoré si vyžadujú adaptácie na úrovni čuchu, zraku, či chuťových buniek. Výsledky teda jasne poukazujú na takéto adaptácie bez toho, aby sme informáciu o takýchto procesoch špecificky vložili do predpokladov výpočtovej analýzy.



Obr. 5.5: Výsledky hľadania pozitívneho výberu v šiestich genómoch cicavcov. Zdroj: Kosiol et al. (2008)

5.4 Zhrnutie

V tejto kapitole sme sa zoznámili so základnými metódami komparatívnej genomiky. Komparatívna genomika sa zaoberá porovnávaním viacerých genómov a pomocou modelov evolúcie sa snaží zistiť novú informáciu o funkciách jednotlivých regiónov týchto genómov. Významnú úlohu hrá prirodzený výber, ktorý zanecháva v evolúcii špeciálne stopy, ktoré je potom zo zarovnaní možno detekovať výpočtovými metódami pomocou rôznych evolučných modelov: ide napríklad o pomalšiu rýchlosť mutácie vo funkčných regiónoch pod vplyvom purifikačného výberu, či prevahu nesynonymných mutácií v prípade pôsobenia pozitívneho výberu.

Komparatívna genomika využíva komplexnú kombináciu metód z pravdepodobnostného modelovania, algoritmov a štatistiky. Zoznámili sme sa tak s technikou nazvanou fylogenetické skryté Markovove modely, s novými modelmi evolúcie a so štatistickým testom pomeru vierohodností.

Literatúra

Alexandersson, M., Cawley, S., and Pachter, L. (2003). SLAM: cross-species gene finding and alignment with a generalized pai r hidden Markov model. *Genome Research*, 13(3):496–502.

Allen, J. E. and Salzberg, S. L. (2005). JIGSAW: integration of multiple sources of evidence for gene prediction. *Bioinformatics*, 21(18):3596–3603.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410.

Altschul, S. F., Madden, T. L., Schaffer, A. A., an d Z. Zhang, J. Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database se arch programs. *Nucleic Acids Research*, 25(17):3389–3392.

Ambros, V., Bartel, B., Bartel, D. P., Burge, C. B., Carrington, J. C., Chen, X., Dreyfuss, G., Eddy, S. R., Griffiths-Jones, S., Marshall, M., Matzke, M., Ruvkun, G., and Tuschl, T. (2003). A uniform system for microRNA annotation. *RNA*, 9(3):277–279.

Andronescu, M., Fejes, A. P., Hutter, F., Hoos, H. H., and Condon, A. (2004). A new algorithm for RNA secondary structure design. *Journal of molecular biology*, 336(3):607–614.

Armen, C. and Stein, C. (1996). A $2\frac{2}{3}$-approximation algorithm for the shortest superstring problem. In *CPM '96: Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching*, pages 87–101, London, UK. Springer-Verlag.

Batzoglou, S., Jaffe, D. B., Stanley, K., Butler, J., Gnerre, S., Mauceli, E., Berger, B., Mesirov, J. P., and Lander, E. S. (2002). ARACHNE: a whole-genome shotgun assembler. *Genome research*, 12(1):177–179.

Birney, E., Clamp, M., and Durbin, R. (2004). GeneWise and Genomewise. *Genome Research*, 14(5):988–995.

Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F. A., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D., Haussler, D., and Miller, W. (2004). Aligning multiple genomic sequences with the threaded blockset aligner. *Genome research*, 14(4):708–715.

Boisserie, J.-R., Lihoreau, F., and Brunet, M. (2005). The position of Hippopotamidae within Cetartiodactyla. *Proceedings of the National Academy of Sciences of the United States of America*, 102(5):1537–1541.

Brejová, B., Brown, D. G., Li, M., and Vinař, T. (2005). ExonHunter: A comprehensive approach to gene finding. *Bioinformatics*, 21(Suppl 1):i57–i65. Proceedings of the 15th International Conference on Inteligent Systems for Molecular Biology (ISMB 2005).

Brejova, B., Brown, D. G., and Vinar, T. (2003). Optimal DNA signal recognition models with a fixed amount of intrasignal dependency. In Benson, G. and Page, R., editors, *Algorithms and Bioinformatics: 3rd International Workshop (WABI)*, volume 2812 of *Lecture Notes in Bioinformatics*, pages 78–94, Budapest, Hungary. Springer.

Brejova, B. and Vinar, T. (2002). A better method for length distribution modeling in HMMs and its application to gene finding. In Apostolico, A. and Takeda, M., editors, *Combinatorial Pattern Matching, 13th Annual Symposium (CPM)*, volume 2373 of *Lecture Notes in Computer Science*, pages 190–202, Fukuoka, Japan. Springer.

Brown, A. H. (1975). Sample sizes required to detect linkage disequilibrium between two or three loci. *Theoretical Population Biology*, 8(2):184–201.

Burge, C. B. and Karlin, S. (1997). Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78–94.

Chalk, A. M., Wahlestedt, C., and Sonnhammer, E. L. L. (2004). Improved and automated prediction of effective siRNA. *Biochemical and biophysical research communications*, 319(1):264–264.

Chor, B. and Tuller, T. (2005). Maximum likelihood of evolutionary trees is hard. In *RECOMB 2005: Research in Computational Biology*, volume 3500 of *Lecture Notes in Computer Science*, pages 296–310.

Clamp, M., Fry, B., Kamal, M., Xie, X., Cuff, J., Lin, M. F., Kellis, M., Lindblad-Toh, K., and Lander, E. S. (2007). Distinguishing protein-coding and noncoding genes in the human genome. *Proc Natl Acad Sci U S A*, 104(49):19428–33.

Clark, A. G. (1990). Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular biology and evolution*, 7(2):111–112.

Cooper, G. M. and Hausman, R. E. (2004). *The Cell: A Molecular Approach, 3rd ed.* Sinauer.

Dawkins, R. (2004). *The Ancestor's Tale.* Mariner Books.

DeCaprio, D., Vinson, J. P., Pearson, M. D., Montgomery, P., Doherty, M., and Galagan, J. E. (2007). Conrad: gene prediction using conditional random fields. *Genome research*, 17(9):1389–1398.

Delcher, A. L., Harmon, D., Kasif, S., White, O., and Salzberg, S. L. (1999). Improved microbial gene identification with GLIMMER. *Nucleic acids research*, 27(23):4636–4641.

Dembo, A., Karlin, S., and Zeitouni, O. (1994). Limit distributions of maximal non-aligned two-sequence segmental score. *The Annals of Probability*, 22:2022–2039.

Do, C. B., Woods, D. A., and Batzoglou, S. (2006). CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–98.

Dowell, R. D. and Eddy, S. R. (2004). Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC bioinformatics*, 5:71.

Durbin, R., Eddy, S., Krogh, A., and Mitchison, G. (1998). *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press.

Edgar, R. C. (2004). MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC bioinformatics*, 5:113.

Elias, I. (2006). Settling the intractability of multiple alignment. *Journal of Computational Biology*, 13(7):1323–1329.

Felsenstein, J. (2004). *Inferring Phylogenies*. Sinauer.

Gibbs, R. A. et al. (2007). Evolutionary and biomedical insights from the rhesus macaque genome. *Science*, 316(5822):222–224.

Goldman, N. and Yang, Z. (1994). A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Molecular biology and evolution*, 11(5):725–726.

Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S. R., and Bateman, A. (2005). Rfam: annotating non-coding RNAs in complete genomes. *Nucleic acids research*, 33(Database issue):D121–124.

Gross, S. S. and Brent, M. R. (2006). Using multiple alignments to improve gene prediction. *Journal of computational biology : a journal of computational molecular cell biology*, 13(2):379–383.

Gross, S. S., Do, C. B., Sirota, M., and Batzoglou, S. (2007). CONTRAST: a discriminative, phylogeny-free approach to multiple informant de novo gene prediction. *Genome biology*, 8(12):R269.

Guigo, R. et al. (2006). EGASP: the human ENCODE Genome Annotation Assessment Project. *Genome Biology*, 7 Suppl 1:1–31.

Gusfield, D. (2002). Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In *RECOMB '02: Proceedings of the sixth annual international conference on Computational biology*, pages 166–175, New York, NY, USA. ACM Press.

Gusfield, D. (2003). Haplotype Inference by Pure Parsimony. In *Combinatorial Pattern Matching (CPM 2003)*, Lecture Notes on Computer Science, pages 144–155. Springer.

Hartl, D. L. (2000). *A Primer of Population Genetics (3rd ed)*. Sinauer Associates.

Hasegawa, M., Kishino, H., and Yano, T. (1985). Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of molecular evolution*, 22(2):160–164.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A*, 89(22):10915–9.

Higgins, D. G., Thompson, J. D., and Gibson, T. J. (1996). Using CLUSTAL for multiple sequence alignments. *Methods in enzymology*, 266:383–402.

Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343.

Hofacker, I. L. (2003). Vienna RNA secondary structure server. *Nucleic acids research*, 31(13):3429–3431.

Kaplan, H. and Shafrir, N. (2005). The greedy algorithm for shortest superstrings. *Information Processing Letters*, 93(1):13–17.

Karlin, S. and Altschul, S. F. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc Natl Acad Sci U S A*, 87(6):2264–2268.

Kent, W. J. (2002). BLAT–the BLAST-like alignment tool. *Genome Research*, 12(4):656–664.

Kent, W. J., Baertsch, R., Hinrichs, A., Miller, W., and Haussler, D. (2003). Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Proceedings of the National Academy of Sciences of the United States of America*, 100(20):11484–9.

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., and Haussler, D. (2002). The human genome browser at UCSC. *Genome Research*, 12(6):996–1006.

Knudsen, B. and Hein, J. (2003). Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic acids research*, 31(13):3423–3428.

Korf, I., Flicek, P., Duan, D., and Brent, M. R. (2001). Integrating genomic homology into gene structure prediction. *Bioinformatics*, 17(S1):S140–148. ISMB 2001.

Kosiol, C., Vinar, T., da Fonseca, R. R., Hubisz, M. J., Bustamante, C. D., Nielsen, R., and Siepel, A. (2008). Patterns of positive selection in six Mammalian genomes. *PLoS Genet*, 4(8):e1000144.

Krogh, A. (1997). Two methods for improving performance of an HMM and their application for gene finding. In *Proceedings of the fifth International Conference on Intelligent Systems for Molecular Biology (ISMB 1997)*, pages 179–186.

Lai, E. C., Tomancak, P., Williams, R. W., and Rubin, G. M. (2003). Computational identification of Drosophila microRNA genes. *Genome biology*, 4(7):R42.

Lin, M. F., Carlson, J. W., Crosby, M. A., Matthews, B. B., Yu, C., Park, S., Wan, K. H., Schroeder, A. J., Gramates, L. S., St Pierre, S. E., Roark, M., Wiley Jr., K. L., Kulathinal, R. J., Zhang, P., Myrick, K. V., Antone, J. V., Celniker, S. E., Gelbart, W. M., and Kellis, M. (2007). Revisiting the protein-coding gene catalog of Drosophila melanogaster using 12 fly genomes. *Genome Res*, 17(12):1823–1826.

Lukashin, A. V. and Borodovsky, M. (1998). GeneMark.hmm: new solutions for gene finding. *Nucleic acids research*, 26(4):1107–1115.

Lyngso, R. B. and Pedersen, C. N. (2000). RNA pseudoknot prediction in energy-based models. *Journal of computational biology*, 7(3-4):409–417.

Majoros, W. H., Pertea, M., and Salzberg, S. L. (2004). TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders. *Bioinformatics*, 20(16):2878–2879.

Margulies, E. H. et al. (2007). Analyses of deep mammalian sequence alignments and constraint predictions for 1% of the human genome. *Genome research*, 17(6):760–764.

Mathews, D. H., Disney, M. D., Childs, J. L., Schroeder, S. J., Zuker, M., and Turner, D. H. (2004). Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences of the United States of America*, 101(19):7287–7292.

Matzke, M. A. and Matzke, A. J. M. (2004). Planting the seeds of a new paradigm. *PLoS biology*, 2(5):E133.

Mills, R. E., Luttig, C. T., Larkins, C. E., Beauchamp, A., Tsui, C., Pittard, W. S., and Devine, S. E. (2006). An initial map of insertion and deletion (INDEL) variation in the human genome. *Genome research*, 16(9):1182–1190.

Mitrophanov, A. Y. and Borodovsky, M. (2006). Statistical significance in biological sequence analysis. *Brief Bioinform*, 7(1):2–24.

Mouse Genome Sequencing Consortium (2002). Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915):520–522.

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

Niu, T., Qin, Z. S., Xu, X., and Liu, J. S. (2002). Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *American journal of human genetics*, 70(1):157–159.

Nussinov, R., Piecznik, G., Grigg, J., and Kleitman, D. (1978). Algorithms for loop matchings. *SIAM Journal on Applied Mathematics*, 35:68–82.

Ohler, U., Yekta, S., Lim, L. P., Bartel, D. P., and Burge, C. B. (2004). Patterns of flanking sequence conservation and a characteristic upstream motif for microRNA gene identification. *RNA*, 10(9):1309–1312.

O'Leary, M. A. and Gatesy, J. (2008). Impact of increased character sampling on the phylogeny of cetartiodactyla (mammalia): combined analysis including fossils. *Cladistics*, 24:397–442.

Pace, N. R. (1997). A molecular view of microbial diversity and the biosphere. *Science*, 276(5313):734–740.

Pasvol, G., Weatherall, D. J., and Wilson, R. J. (1978). Cellular mechanism for the protective effect of haemoglobin S against P. falciparum malaria. *Nature*, 274(5672):701–703.

Pearson, W. R. and Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85(8):2444–2448.

Pedersen, J. S., Bejerano, G., Siepel, A., Rosenbloom, K., Lindblad-Toh, K., Lander, E. S., Kent, J., Miller, W., and Haussler, D. (2006). Identification and classification of conserved RNA secondary structures in the human genome. *PLoS Comput Biol*, 2(4):e33.

Pertea, M. and Salzberg, S. L. (2010). Between a chicken and a grape: estimating the number of human genes. *Genome Biol*, 11(5):206.

Pevzner, P. A., Tang, H., and Waterman, M. S. (2001). An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences of the United States of America*, 98(17):9748–9753.

Pritchard, J. K., Stephens, M., and Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–949.

Rabiner, L. R. (1989). A tutorial on Hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

Ren, J., Rastegari, B., Condon, A., and Hoos, H. H. (2005). HotKnots: heuristic prediction of RNA secondary structures including pseudoknots. *RNA*, 11(10):1494–1494.

Rivas, E. and Eddy, S. R. (1999). A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of molecular biology*, 285(5):2053–2058.

Roberts, L. (2001). The human genome. Controversial from the start. *Science*, 291(5507):1182–1188.

Rosenberg, N. A., Pritchard, J. K., Weber, J. L., Cann, H. M., Kidd, K. K., Zhivotovsky, L. A., and Feldman, M. W. (2002). Genetic structure of human populations. *Science*, 298(5602):2381–2385.

Sankoff, D. (1985). Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM Journal on Applied Mathematics*, 45(5):810–825.

Sethupathy, P., Megraw, M., and Hatzigeorgiou, A. G. (2006). A guide through present computational approaches for the identification of mammalian microRNA targets. *Nature methods*, 3(11):881–886.

Shao, W., Tang, J., Song, W., Wang, C., Li, Y., Wilson, C. M., and Kaslow, R. A. (2007). CCL3L1 and CCL4L1: variable gene copy number in adolescents with and without human immunodeficiency virus type 1 (HIV-1) infection. *Genes and immunity*, 8(3):224–231.

Siepel, A., Bejerano, G., Pedersen, J. S., Hinrichs, A. S., Hou, M., Rosenbloom, K., Clawson, H., Spieth, J., Hillier, L. W., Richards, S., Weinstock, G. M., Wilson, R. K., Gibbs, R. A., Kent, W. J., Miller, W., and Haussler, D. (2005). Evolutionarily conserved

elements in vertebrate, insect, worm, and yeast genomes. *Genome research*, 15(8):1034–1040.

Siepel, A. and Haussler, D. (2004). Computational identification of evolutionarily conserved exons. In *RECOMB 2004: Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology*, pages 177–186. ACM Press.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

Solovyev, V., Kosarev, P., Seledsov, I., and Vorobyev, D. (2006). Automatic annotation of eukaryotic genes, pseudogenes and promoters. *Genome biology*, 7 Suppl 1:.1–12.

Stanke, M., Schoffmann, O., Morgenstern, B., and Waack, S. (2006). Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. *BMC bioinformatics*, 7:62.

Stanke, M. and Waack, S. (2003). Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, 19(S2):II215–II225. ECCB 2003.

Stephens, M., Smith, N. J., and Donnelly, P. (2001). A new statistical method for haplotype reconstruction from population data. *American journal of human genetics*, 68(4):978–979.

Studier, J. A. and Keppler, K. J. (1988). A note on the neighbor-joining algorithm of Saitou and Nei. *Molecular biology and evolution*, 5(6):729–731.

Surridge, A. K., Osorio, D., and Mundy, N. I. (2003). Evolution and selection of trichromatic vision in primates. *Trends in Ecology and Evolution*, 18(4).

Sutter, N. B., Bustamante, C. D., Chase, K., Gray, M. M., Zhao, K., Zhu, L., Padhukasahasram, B., Karlins, E., Davis, S., Jones, P. G., Quignon, P., Johnson, G. S., Parker, H. G., Fretwell, N., Mosher, D. S., Lawler, D. F., Satyaraj, E., Nordborg, M., Lark, K. G., Wayne, R. K., and Ostrander, E. A. (2007). A single IGF1 allele is a major determinant of small size in dogs. *Science*, 316(5821):112–115.

Sweedyk, Z. (2000). A 21/2-approximation algorithm for shortest superstring. *SIAM Journal on Computing*, 29(3):954–986.

The International HapMap Consortium (2005). A haplotype map of the human genome. *Nature*, 437(7063):1299–1300.

Washietl, S., Hofacker, I. L., and Stadler, P. F. (2005). Fast and reliable prediction of noncoding RNAs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(7):2454–2459.

Weinberg, Z. and Ruzzo, W. L. (2004). Exploiting conserved structure for faster annotation of non-coding RNAs without loss of accuracy. *Bioinformatics*, 20 Suppl 1:I334–I341.

Y. Benjamini, Y. H. (1995). Controlling the False Discovery Rate: a Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B*, 57(1):289–300.

Yang, Z. and Nielsen, R. (2002). Codon-substitution models for detecting molecular adaptation at individual sites along specific lineages. *Molecular biology and evolution*, 19(6):908–917.

Zhang, J., Nielsen, R., and Yang, Z. (2005). Evaluation of an improved branch-site likelihood method for detecting positive selection at the molecular level. *Molecular biology and evolution*, 22(12):2472–2479.

Zuker, M. (2003). Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic acids research*, 31(13):3406–3415.

Zuker, M. and Stiegler, P. (1981). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1):133–138.