Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Rekonštrukcia duplikačných histórií pomocou pravdepodobnostného modelu

Bakalárska práca

2014

Ján Hozza

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Rekonštrukcia duplikačných histórií pomocou pravdepodobnostného modelu

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 2508 Informatika
Školiace pracovisko: Katedra Informatiky FMFI
Vedúci práce: Mgr. Tomáš Vinař, PhD.

Bratislava, 2014

Ján Hozza

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Ján Hozza
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Rekonštrukcia duplikačných histórií pomocou pravdepodobnostného modelu
Reconstruction of Duplication Histories using a Probabilistic Model

Cieľ: V rámci evolúcie DNA sekvencií nastávajú tzv. duplikácie, kde sa segment DNA skopíruje na iné miesto poblíž zdroja. Duplikácie sa tiež kombinujú s operáciami menšieho rozsahu (substitúcie, delécie, inzercie), čím vzniká sekvencia so zložitou štruktúrou. Cieľom práce je navrhnúť a implementovať algoritmus na rekonštrukciu evolučnej histórie duplikovanej sekvencie na základe existujúceho pravdepodobnostného modelu. Oproti predchádzajúcim prácam v tejto oblasti pôjde najmä o vylepšenie optimalizačného a navrhovacieho algoritmu, ako aj o preskúmanie účinnosti rôznych možností skórovania jednotlivých udalostí pri analýze simulovaných a reálnych dát.

Vedúci: Mgr. Tomáš Vinař, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.

Dátum zadania: 20.10.2013

Dátum schválenia: 21.10.2013 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie

Chcel by som poďakovať najmä môjmu vedúcemu Tomášovi Vinařovi za trpezlivosť, cenné rady a obetovaný čas a Broni Brejovej za pomoc s testovacími dátami. Moja vďaka patrí aj bioinformatickej komunite za poskytnutie výpočtovej techniky a stovky hodín procesorového času. Ďakujem tiež rodine a priateľom za podporu.

Abstrakt

V bioinformatike sa často zaoberáme evolúciou DNA sekvencie jedného organizmu. Pre danú sekvenciu nás môže zaujímať duplikačná história, čo je postupnosť dlhých zmien v sekvencii. V práci sa zaoberáme problémom ako zrekonštruovať duplikačnú históriu, ak poznáme len súčasnú podobu DNA sekvencie. Implementujeme algoritmus riešiaci tento problém pomocou pravdepodobnostného modelu a šiestich skórovacích metód. Hlavným prínosom našej práce je samotná implementácia a tiež podrobné preskúmanie a porovnanie vplyvu skórovacích metód na výsledky algoritmu.

Kľúčové slová: duplikačná história, rekonštrukcia, pravdepodobnostný model, skórovanie

Abstract

In bioinformatics, we often study the evolution of DNA sequences. In this work we are especially interested in duplication history, which is a list of long changes in the sequence. We focus on the problem of reconstruction of duplication history given the current form of the sequence. We implement an algorithm solving this problem using a probabilistic model and six scoring methods. The main contribution of our work is the implementation itself and also a detailed examination and comparison of impact of the scoring methods on the results of the algorithm.

Keywords: duplication history, reconstruction, probabilistic model, scoring

Obsah

Úvod **1**

1 Biologické pozadie problému **2**
1.1 DNA sekvencia 2
1.2 Zarovnania DNA sekvencií 2
1.3 Vývoj DNA 3
1.4 Duplikačná história 4
1.5 Lokus a strom lokusu 6

2 Pravdepodobnostný model **8**
2.1 Udalosti 8
2.2 Mutácie 9
2.3 Evolúcia sekvencie 10
2.4 Výpočet pravdepodobnosti histórie 10

3 Algoritmus na rekonštrukciu histórie **12**
3.1 Atomizácia sekvencie 13
3.2 Neodhaliteľnosť niektorých udalostí 15
3.3 Evolučné stromy pre lokusy a atómy 16
3.4 Zrýchlenie výpočtu vierohodnosti 16
3.5 Predchádzajúce prístupy k rekonštrukcii histórie 18
3.6 Vzorkovanie duplikačnej udalosti pomocou DP 18
3.7 Náš algoritmus na rekonštrukciu histórie 19
3.8 Určovanie časov udalostí 21
3.9 Formát vstupu a výstupu 22

4 Skórovanie duplikačných udalostí **24**
4.1 Dĺžka duplikácie 25
4.2 Podobnosť sekvencií 25
4.3 Čiastočná duplikácia 26

4.4 Zníženie rôznorodosti atómov 26
4.5 Výskyt v predošlých históriách 27
4.6 Dobrá minulosť 27

5 Testovanie algoritmu, porovnania a výsledky **29**
5.1 Čas behu programu 30
5.2 Spôsob testovania a hodnotenia úspešnosti 31
5.3 Výsledky testovania 32

Záver **37**

Zoznam obrázkov

1.1 Príklad zarovnania DNA sekvencií 3
1.2 Príklad histórie 5
1.3 Fylogenetický strom človeka a niektorých príbuzných druhov. 5
1.4 Príklad stromu lokusu 6

3.1 Sekvencie v histórii rozdelené na atómy 14
3.2 Príklad histórie sekvencie atómov 15
3.3 Ukážka grafu pre postupnosť `1 2 -1 2 1 2 -1` 20
3.4 Príklad súboru so zoznamom atómov 22
3.5 Príklad súboru so zarovnaniami 1. skupiny atómov. 22
3.6 Príklad výstupného súboru 23

5.1 Porovnanie rýchlostí výpočtu pravdepodobnosti histórie 30
5.2 Porovnanie rýchlostí metód skórovania 31
5.3 Závislosť úspešnosti metód od váhy pre vstup A. 32
5.4 Závislosť úspešnosti metód od váhy pre vstup C. 34
5.5 Víťazné nastavenia váh otestované na všetkých vstupoch. 36

Úvod

V snahe zistiť čo najviac o ľudskom organizme skúmajú biológovia DNA sekvencie našich buniek. Avšak tieto sekvencie sú príliš dlhé na to, aby sa dali skúmať ručne. Vznikla potreba začleniť do tohoto skúmania počítače. Pre tieto účely vznikla bioinformatika, veda ktorá využíva informatiku na spracovávanie veľkého množstva dát z biológie.

V našej práci skúmame vývoj, resp. evolúciu DNA sekvencií. Zaoberáme sa najmä dlhými udalosťami, čiže dlhými duplikáciami, dlhými duplikáciami s inverziou a dlhými deléciami. Nadväzujeme na práce Vinaňa et al. [VBSS10] a Kravca [Kra11], ktoré sa touto témou už zaoberali.

Problém rekonštrukcie duplikačnej histórie pre danú DNA sekvenciu je nájsť postupnosť udalostí, ktoré túto sekvenciu vytvorili. Keďže vyriešiť problém exaktne je ťažké a správna postupnosť udalostí sa nedá jednoznačne určiť, používajú sa na riešenie pravdepodobnostné metódy. V našej práci sa zoznámime s týmto problémom, predstavíme si algoritmus, ktorý tento problém rieši a preskúmame vplyv šiestich parametrov na výsledky algoritmu. Porovnáme aj dva druhy výpočtu pravdepodobnosti histórií.

V prvej kapitole sa zoznámime so základnými pojmami, ktoré budeme v práci často používať a ukážeme si biologickú stránku celého problému. V druhej kapitole si predstavíme pravdepodobnostný model, ktorý má v algoritme významnú úlohu. V tretej a štvrtej kapitole sa budeme venovať samotnému algoritmu. Spomenieme predchádzajúcu prácu v tejto oblasti, a tiež popíšeme náš algoritmus a niektoré jeho dôležité súčasti. Podrobnejšie rozoberieme šesť skórovacích metód. V piatej kapitole otestujeme algoritmus, preskúmame vplyv parametrov na jeho efektivitu a úspešnosť.

Zdrojové súbory algoritmu a použité testovacie dáta sa dajú nájsť na webovej stránke `http://people.ksp.sk/~janoh/bakalarka.php`.

Kapitola 1

Biologické pozadie problému

1.1 DNA sekvencia

Dôležitou súčasťou všetkých mnohobunkových organizmov na Zemi je deoxyribonukleová kyselina, skrátene DNA. Táto DNA sa nachádza v chromozómoch v jadrách eukaryotických buniek vo forme dvojzávitnice. Dvojzávitnica sa skladá z dvoch komplementárnych vlákien, pričom každé z vlákien je reťazec nukleotidov. Nukleotid je chemická molekula, ktorá obsahuje jednu zo štyroch báz adenín, cytozín, guanín alebo timín.

Pre informatikov nie je chemická štruktúra nukleotidov veľmi zaujímavá, preto si môžeme DNA sekvencie abstraktne predstaviť ako reťazec písmen z abecedy {`A`,`C`,`G`,`T`}. Ten sa zo skutočnej sekvencie získava sekvenovaním. My budeme predpokladať, že už reťazec poznáme, a to dokonca veľmi presne. Keď teda budeme v tejto práci hovoriť o DNA sekvencii, budeme mať na mysli postupnosť písmen.

1.2 Zarovnania DNA sekvencií

Veľmi často nás pri skúmaní DNA zaujímajú rovnaké alebo podobné úseky. Totiž, keď sa nejaké dosť dlhé úseky veľmi podobajú, tak skoro určite vznikli z nejakého spoločného predka. Je dosť nepravdepodobné, že by také dlhé podobné sekvencie vznikli nezávisle. Viac si o tom povieme v podkapitole 3.1.

Nie je prekvapivé, že DNA sekvencia konkrétneho človeka sa bude veľmi podobať na DNA sekvenciu jeho bratranca. Tiež, keď porovnáme ľudskú DNA s DNA šimpanza, tak nájdeme mnoho podobných úsekov. Podobnosť objavíme aj pri vzdialenejších živočíšnych druhoch, no bude čím ďalej, tým ťažšie viditeľná.

Pre nás je zaujímavý iný typ podobnosti, podobnosť v rámci jednej sekvencie. Skúmame, prečo v DNA sekvencii organizmu, napríklad človeka, často nachádzame obrov-

ské kusy DNA, dlhé desaťtisíce báz, ktoré sa na seba výrazne podobajú.

Na vyjadrenie a popísanie toho, že si dva úseky DNA nejakým spôsobom zodpovedajú, t.j. majú rovnakého predka, používame zarovnania. Zarovnanie je zápis dvoch alebo viacerých úsekov DNA pod seba, s doplnenými pomlčkami medzi niektoré písmená tak, aby súčet skóre všetkých dvojíc znakov nad sebou bol čo najvyšší. Napríklad skóre dvojice rovnakých písmen môže byť 1, skóre rôznych písmen -1, skóre pomlčky s hocičím -3. Príklad zarovnania reťazcov `TTCGTCGAAACTTGACCGAT` a `TACGAATAGAAACATGACGAT` môžeme vidieť na obrázku 1.1.

```
TTCG--TCGAAACTTGACCGAT
TACGAATAGAAACATGA-CGAT
```

Obr. 1.1: Príklad zarovnania DNA sekvencií

1.3 Vývoj DNA

Dôležitá vlastnosť sekvencií je možnosť kopírovať ju. Bez tejto vlastnosti by neboli organizmy schopné rozmnožovať sa. Počas kopírovania DNA môžu nastať rôzne typy chýb, ktoré pôvodnú DNA sekvenciu skopírujú trocha odlišne. Vďaka týmto chybám sa sekvencia postupne vyvíja.

Najčastejší typ chýb sú krátke zmeny, ktoré sa týkajú zväčša jedného písmena. Tieto zmeny nazývame mutácie a rozdeľujeme ich na tri typy:

- Substitúcia – jedno písmeno v reťazci sa zmení na iné.
- Krátka inzercia – do reťazca sa vloží zopár písmen.
- Krátka delécia – v reťazci sa zmaže zopár písmen.

Prítomnosť týchto mutácií najlepšie vidíme v zarovnaniach. Nás budú zaujímať hlavne substitúcie. Pomlčky v zarovnaniach budeme skôr intepretovať ako chýbajúce dáta, čiže ako keby tam mohlo byť ľubovoľné písmeno.

Druhý typ chýb v DNA sú dlhé zmeny, ktoré sa týkajú dlhých kusov DNA. Sú oveľa zriedkavejšie ako mutácie a ich dĺžky sú obvykle tisíce až desaťtisíce báz.

- Duplikácia – nejaký kus reťazca sa skopíruje na iné miesto v reťazci.
- Duplikácia s inverziou – nejaký kus reťazca sa skopíruje na iné miesto v reťazci a prevráti sa.

- Dlhá delécia – zmaže sa nejaký kus reťazca.

Spoločný názov pre tieto tri zmeny bude udalosť. Definujme si prevrátenie alebo inverziu reťazca S, že je to komplement reverzu S. Reverz je napísanie reťazca odzadu a komplement je paralelné nahradenie všetkých výskytov `A` za `T`, `T` za `A`, `C` za `G` a `G` za `C`. Napríklad inverzia sekvencie `AAAGCATC` je `GATGCTTT`.

Na presné popísanie udalosti potrebujeme tieto údaje:

- Čas, kedy udalosť nastala.
- Typ udalosti, či je to duplikácia, duplikácia s inverziou alebo delécia.
- Pozícia udalosti – za koľkým písmenom v sekvencii nasleduje duplikovaná resp. vymazaná časť sekvencie.
- Dĺžka udalosti – aká dlhá je duplikovaná resp. vymazaná časť sekvencie.
- V prípade duplikácií potrebujeme vedieť aj vzdialenosť medzi pôvodnou a cieľovou pozíciou udalosti.

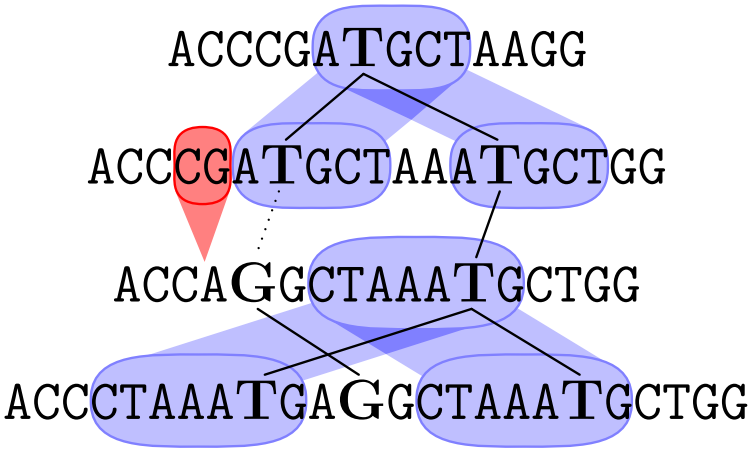
1.4 Duplikačná história

Duplikačná história alebo evolučná história duplikovanej sekvencie, ktorú budeme často nazývať skrátene história, je postupnosť udalostí nejakej DNA sekvencie. K histórii obvykle pridávame špeciálnu udalosť konca, ktorej jediný parameter je čas. Čas tejto udalosti je súčasnosť a slúži na vyjadrenie faktu, že od poslednej duplikácie alebo delécie po súčasnosť nenastala žiadna takáto udalosť.

V histórii na obrázku 1.2 bola na začiatku sekvencia `ACCGATGCTAAGG`, v ktorej nastala duplikácia piatich báz `ATGCT`. Potom sa udiala delécia písmen `CG` a napokon nastala duplikácia `CTAAATG`. Niekedy medzi deléciou a druhou duplikáciou sa udiala jedna mutácia, jedno `T` sa zmenilo na `G`.

Osekvenovaním DNA nejakého organizmu vieme zistiť, ako vyzerá sekvencia v súčasnosti, čiže najnižší riadok na obrázku. Našim cieľom je navrhnúť a implementovať algoritmus, ktorý sa pokúsi z takejto sekvencie zrekonštruovať jej históriu, teda zistiť, aké udalosti nastali.

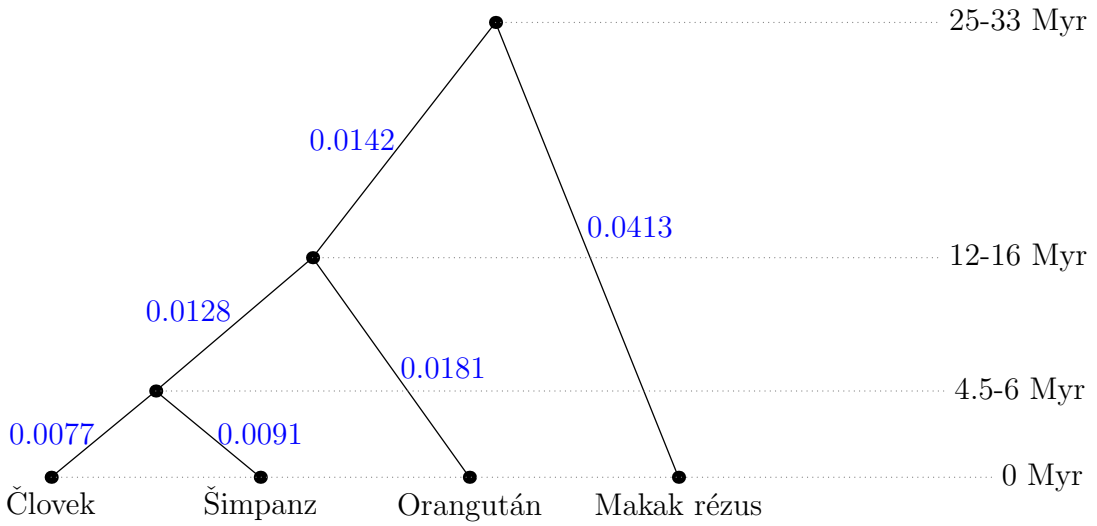
Vzhľadom na to, že rôzne postupnosti udalostí môžu viesť k rovnakej koncovej sekvencii, nevieme určiť históriu jednoznačne. Preto budeme hľadať najpravdepodobnejšiu zo všetkých možných histórií. Našou snahou bude navrhnúť výpočet pravdepodobnosti histórie tak, aby v praxi tá najpravdepodobnejšia bola tá správna.



Obr. 1.2: Krátky príklad histórie bez určených časov. Udalosti sú postupne duplikácia, delécia a duplikácia. Zvýraznená je evolúcia jedného znaku, v ktorom nastala aj substitúcia.

Pred hľadaním histórie si tiež musíme určiť počiatočný čas, od ktorého nás história zaujíma. Čím skorší čas si určíme (čím ďalej od súčasnosti), tým ťažšie sa nám bude hľadať história, ale tým viac udalostí môžeme nájsť. Tento čas bude mať hodnotu $t_0 = 0$ a čas udalosti E bude mať hodnotu $t(E)$ v jednotkách času od času t_0.

Jednotku času definujeme na základe rýchlosti mutácií podľa pravdepodobnostného modelu v 2. kapitole. Na obrázku 1.3 sú pre spoločných predkov človeka a vybranej skupiny príbuzných organizmov znázornené približné časy v miliónoch rokoch (Myr) pred našim letopočtom [Ora11]. Časy na hranách sú udávané v jednotkách času a boli vypočítané na základe zarovnania jedného ľudského chromozómu k zvyšným organizmom.



Obr. 1.3: Fylogenetický strom človeka a niektorých príbuzných druhov.

Keby sme chceli skonštruovať históriu ľudskej DNA sekvencie od času spoločného predka človeka a šimpanza, čas koncovej udalosti by na základe obrázku 1.3 bol 0.0077.

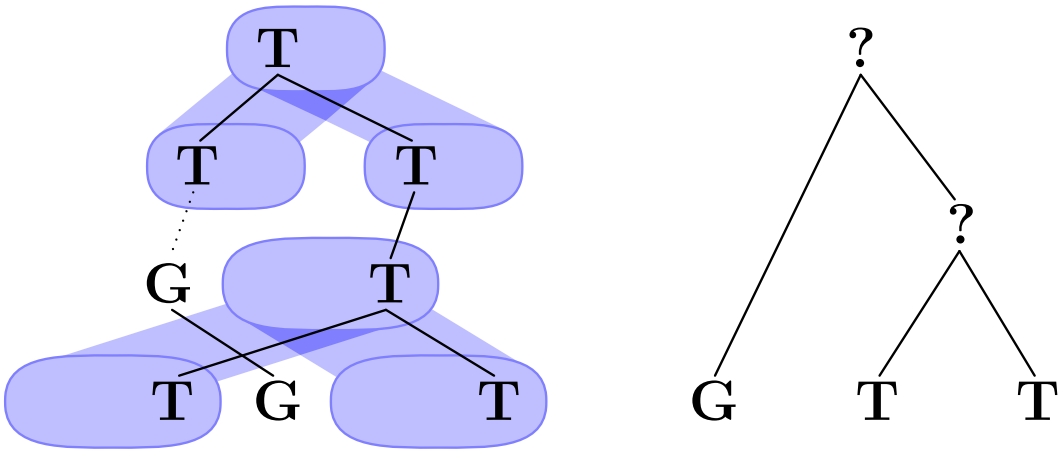
1.5 Lokus a strom lokusu

Jedno písmeno v pôvodnej sekvencii nazývame lokus. Písmená, ktoré vzniknú z tohto písmena budeme považovať za ten istý lokus, len iné jeho inštancie. Ukážme si to na príklade. Nech AACG je pôvodná sekvencia. Nachádzajú sa v nej 4 lokusy, ktorých čísla napíšeme do dolného indexu, $\mathtt{A}_1\mathtt{A}_2\mathtt{C}_3\mathtt{G}_4$. Pri duplikácii písmenka vzniknú dve kópie toho istého lokusu, čiže ak v $\mathtt{A}_1\mathtt{A}_2\mathtt{C}_3\mathtt{G}_4$ nastane duplikácia druhého a tretieho lokusu, môže vzniknúť $\mathtt{A}_1\mathtt{A}_2\mathtt{C}_3\mathtt{A}_2\mathtt{C}_3\mathtt{G}_4$. Takisto lokus ostane tým istým lokusom aj po mutácii, čiže by mohlo vzniknúť $\mathtt{A}_1\mathtt{T}_2\mathtt{C}_3\mathtt{A}_2\mathtt{C}_3\mathtt{C}_4$.

V prípade, že nastane inverzná duplikácia, invertované lokusy budeme písať s opačným znamienkom. Z poslednej sekvencie mohla napríklad inverznou duplikáciou postupnosti lokusov $\mathtt{A}_1\mathtt{T}_2\mathtt{C}_3$ vzniknúť $\mathtt{A}_1\mathtt{T}_2\mathtt{C}_3\mathtt{A}_2\mathtt{C}_3\mathtt{G}_{-3}\mathtt{A}_{-2}\mathtt{T}_{-1}\mathtt{C}_4$

Ak poznáme históriu sekvencie, môžeme sa pozrieť na evolúciu každého lokusu zvlášť. Vyčlenením jedného lokusu dostaneme takzvaný strom lokusu, čo je zakorenený binárny strom, ktorého vrcholy sú rôzne inštancie lokusu, v rôznych časoch.

Samotný tvar stromu zodpovedá evolúcii tohoto lokusu, čiže napríklad vrchol má dvoch synov, pokiaľ nastala duplikácia, ktorá skopírovala tento lokus. Koreň tohto stromu je pôvodná inštancia lokusu. Listy stromu na najnižšej úrovni zodpovedajú výskytom lokusov v súčasnej sekvencii. Listy sa môžu vyskytovať aj na vyšších úrovniach, to zodpovedá vymazaniu inštancie lokusu zo sekvencie, napríklad z dôvodu krátkej alebo dlhej delécie. Dĺžka hrany u–v je rovná rozdielu časov inštancie v a inštancie u.



Obr. 1.4: Strom pre siedmy lokus, podľa histórie na obrázku 1.2. Vľavo so znázornenými duplikáciami. Vpravo zjednodušený a bez informácií o mutáciách.

Na obrázku 1.4 vidíme príklad na strom lokusu. Zo súčasnej histórie dokážeme určiť iba najnižšiu vrstvu tohto stromu. Na to, aby sme zistili znaky v ostatných vrcholoch, by sme potrebovali vedieť kedy a aké mutácie nastali. Z histórie vieme jednoznačne určiť tvary stromov všetkých lokusov, čo si ukážeme v podkapitole 3.3.

Kapitola 2

Pravdepodobnostný model

V tejto kapitole sa budeme zaoberať generatívnym pravdepodobnostným modelom evolúcie sekvencie.

Rýchlosť mutácií môže závisieť od organizmu, dĺžky jeho generácie, vonkajších podmienok, atď. Aby sme sa tým nemuseli zaoberať, abstrahujeme od skutočného času a stanovíme si jednotku času tak, aby priemerný počet mutácii na jednu bázu v sekvencii, za jednotku času, bol jedna. Ďalej budeme predpokladať, že rýchlosť mutácií aj udalostí je konštantná.

2.1 Udalosti

Prvá časť modelu sa zaoberá modelovaním dlhých chýb, teda udalostí. Na určenie udalostí, ako sme už spomínali v kapitole 1, potrebujeme vedieť kedy nastali, ich typ, dĺžku, pozíciu a v prípade duplikácií aj vzdialenosť. Každú vlastnosť si predstavíme ako náhodnú premennú s určitým pravdepodobnostným rozdelením. Pravdepodobnostné rozdelenia a hodnoty ich parametrov sme sa rozhodli použiť také, aké popisujú Vinař et al. [VBSS10].

Typ udalosti je s pravdepodobnosťou $p_{del} = 0.05$ delécia, s pravdepodobnosťou $(1-p_{del})(1-p_{inv})$ obyčajná duplikácia a s pravdepodobnosťou $(1-p_{del})\,p_{inv}$ duplikácia s inverziou, pre $p_{inv} = 0.39$.

Dĺžka udalosti a vzdialenosť cieľových úsekov pri duplikácii má geometrické rozdelenie so strednou hodnotou $m_{len} = 14\,307$ pre dĺžku a $m_{dist} = 306\,718$ pre vzdialenosť. Pozícia sekvencie má rovnomerné rozdelenie na množine $\{1..\ell\}$, kde ℓ je dĺžka celej sekvencie. Neplatné udalosti, ktoré napríklad prečnievajú cez okraj, pri výbere zahadzujeme.

Výskyt udalostí v čase modelujeme Poissonovým procesom, takže čas medzi dvoma udalosťami vyberáme pomocou exponenciálneho rozdelenia. Rýchlosť tohoto procesu

označíme λ. Pri simulovaných dátach Vinař et al. používali hodnoty $\lambda = 200$ aj $\lambda = 300$, my budeme používať len prvú z nich.

Celkovú pravdepodobnosť duplikácie E v sekvencii dĺžky ℓ, ktorá sa udiala t časových jednotiek po poslednej udalosti, označíme $P(E|\ell, t)$ a vypočítame ju ako súčin pravdepodobností pre každú vlastnosť, resp. hustotu pravdepodobnosti, pokiaľ ide o čas.

$$P(E|\ell, t) = P(t(E)) \cdot p_l(1-p_l)^{l(E)-1} \cdot p_d(1-p_d)^{d(E)} \cdot \frac{1}{\ell} \cdot \lambda e^{-\lambda t} \tag{2.1}$$

Kde $t(E)$, $l(E)$, $d(E)$ sú čas, dĺžka a vzdialenosť duplikácie E, ďalej $p_l = \frac{1}{m_{len}}$ a $p_d = \frac{1}{m_{dist}}$. Pre deléciu dostaneme podobný vzťah, len vynecháme člen týkajúci sa vzdialenosti. Špeciálne pre koncovú udalosť E_h, chceme započítať pravdepodobnosť, že za posledných t časových jednotiek nenastala žiadna udalosť.

$$P(E_h|t) = e^{-\lambda t} \tag{2.2}$$

2.2 Mutácie

Druhá časť pravdepodobnostného modelu bude popisovať, akým spôsobom sa dejú mutácie. Preto vyjadríme, aká je pravdepodobnosť, že sa sekvencia y zmenila na sekvenciu x za čas t, čo budeme označovať $P(x|y, t)$.

Budeme predpokladať, že každá pozícia v sekvencii mutuje nezávisle od ostatných a rovnako rýchlo. Potom nám na popísanie procesu evolúcie stačí vedieť, aké sú pravdepodobnosti $P(a|b, t)$ pre $a, b \in \{A, C, G, T\}$. Tieto hodnoty si môžeme napísať do tzv. substitučnej matice

$$S(t) = \begin{pmatrix} P(A|A,t) & P(A|C,t) & P(A|G,t) & P(A|T,t) \\ P(C|A,t) & P(C|C,t) & P(A|G,t) & P(A|T,t) \\ P(G|A,t) & P(G|C,t) & P(A|G,t) & P(A|T,t) \\ P(T|A,t) & P(T|C,t) & P(A|G,t) & P(A|T,t) \end{pmatrix}$$

Zrejme $\sum_b P(a|b, t_1)\, P(b|c, t_2) = P(a|c, t_1 + t_2)$, takže pre túto maticu musí platiť, že $S(t_1)\, S(t_2) = S(t_1 + t_2)$.

Tu sme sa rozhodli použiť Jukes-Cantorov model, popísaný v knihe od Durbina et. al [DEKM98]. Podľa tohto modelu je mutácia zo znaku a na znak b rovnako pravdepodobná pre všetky dvojice (a, b).

Potom substitučná matica má tvar

$$S(t) = \begin{pmatrix} r_t & s_t & s_t & s_t \\ s_t & r_t & s_t & s_t \\ s_t & s_t & r_t & s_t \\ s_t & s_t & s_t & r_t \end{pmatrix},$$

kde

$$r_t = \tfrac{1}{4}(1 + 3e^{-4\alpha t}),$$
$$s_t = \tfrac{1}{4}(1 - e^{-4\alpha t}).$$

Konštanta α určuje, ako rýchlo sa dejú mutácie. Keďže sme si určili, že za jednotku času sa priemerne udeje jedna mutácia na písmeno, α bude mať hodnotu $\frac{1}{3}$.

2.3 Evolúcia sekvencie

V tejto podkapitole si ukážeme, ako vyzerá evolúcia, podľa vyššie uvedeného substitučného modelu. Na začiatku si model vygenerujeme náhodnú počiatočnú sekvenciu s_0 fixnej dĺžky. Správnu dĺžku sekvencie v počiatočnom čase t_0 dokážeme zistiť zo súčasnej sekvencie postupom, ktorý si ukážeme v podkapitole 3.3.

Následne sa náhodným procesom, s rýchlosťami a pravdepodobnostným rozdelením popísaným vyššie, v sekvencii generujú udalosti a mutácie. Sekvenciu po i udalostiach označíme s_i. Potom, keď nastali všetky udalosti, ktorých počet si označíme h, vznikla výsledná sekvencia $s_h = s$.

2.4 Výpočet pravdepodobnosti histórie

Keď už máme vybudovaný model, môžeme ho použiť pri riešení nášho problému, ktorým je hľadanie histórie. Model by sme chceli využiť na to, aby sme pre dve histórie vedeli rozhodnúť, ktorá z nich má vyššiu pravdepodobnosť. Zaujíma nás $P(H, s|\ell_0)$, teda pravdepodobnosť, že model z počiatočnej sekvencie dĺžky ℓ_0 vygeneruje danú históriu H a zároveň skončí so sekvenciou s. Túto pravdepodobnosť, ktorú budeme nazývať pravdepodobnosť histórie, môžeme vypočítať podľa nasledujúceho vzťahu.

$$P(H, s|\ell_0) = P(s|H, \ell_0)\, P(H|\ell_0) \tag{2.3}$$

Prvú časť vypočítame tak, že rozložíme sekvenciu na lokusy. Nech $s^{(i)}$ je podpostupnosť sekvencie s taká, že sme vybrali len písmená i-teho lokusu, pre $1 \leq i \leq \ell_0$. K histórii a dĺžke pôvodnej postupnosti vieme jednoznačne zostrojiť zodpovedajúce

stromy lokusov, ako sme spomínali v podkapitole 1.5. Označme tieto stromy postupne T_1 až T_{ℓ_0}. Potom platí

$$P(s|H,\ell_0) = \prod_{i=1}^{\ell_0} P(s^{(i)}|T_1,\ldots,T_{\ell_0}) \tag{2.4}$$

Keďže $P(s^{(i)})$ závisí len od stromu i-teho lokusu, môžeme ďalej upravovať.

$$P(s|H,\ell_0) = \prod_{i=1}^{\ell_0} P(s^{(i)}|T_i) \tag{2.5}$$

Pri znalosti $P(a|b,t)$, pre $a,b \in \{A,C,G,T\}$ vieme spočítať $P(s^{(i)}|T_i)$ známym Felsensteinovým algoritmom. Totiž $P(s^{(i)}|T_i) = L(T_i|s^{(i)})$, čiže vierohodnosť i-teho stromu. Felsensteinov algoritmus je podrobnejšie popísaný v knihe od Durbina et. al [DEKM98] a v Anderleho práci [And14].

Napokon nám ostáva vypočítať, $P(H|\ell_0)$, čo spravíme rozdelením na jednotlivé udalosti. Nech v H postupne nastali udalosti $E_1, E_2, \ldots E_h$. Označme t_i rozdiel medzi časom E_i a časom E_{i-1}, okrem t_1, kedy to bude len E_1. Dĺžku sekvencie po i-tej udalosti označíme ℓ_i. Potom

$$P(H|\ell_0) = \prod_{i=1}^{h} P(E_i|\ell_{i-1}, t_i) \tag{2.6}$$

Pravdepodobnosť na pravej strane už vypočítame jednoducho, na základe vzťahov 2.1 a 2.2.

Kapitola 3

Algoritmus na rekonštrukciu histórie

Základná kostra algoritmu na počítanie duplikačnej histórie je pomerne jednoduchá. Algoritmus vygeneruje určité množstvo relevantných histórií, pre každú z týchto histórií spočíta jej pravdepodobnosť a nakoniec vyberie tú históriu, ktorá mala pravdepodobnosť najväčšiu. Najprv je však potrebné vstupnú sekvenciu predspracovať, rozdeliť ju na takzvané atómy, o ktorých si povieme v podkapitole 3.1.

Históriami DNA sekvencií, ich rekonštruovaním a generovaním sa už zaoberali Vinař et al. [VBSS10] a Kravec [Kra11]. Vinař et al. využili Metropolis-Hastingsov Markov chain Monte Carlo (MCMC) algoritmus na generovanie histórií z určitého pravdepodobnostného rozdelenia popísaného v ich článku. Ich algoritmus vytváral históriu odzadu, čiže od súčasnej sekvencie po pôvodnú. To, čo robil Kravec, bolo veľmi podobné. Jeho príspevkom bol najmä efektívnejší algoritmus na vzorkovanie jednej udalosti. Nevýhodou jeho prístupu bol príliš zjednodušený spôsob akým sa rozhodujeme, ktorú udalosť vybrať. Viac o ich algoritmoch si povieme neskôr v podkapitole 3.5.

V našej práci navrhneme algoritmus, ktorý spája výhody oboch predošlých postupov. Využívame vzorkovanie z Kravcovho algoritmu, vďaka čomu je náš algoritmus rýchly. Používame zložené skórovanie udalostí podobné tomu čo použil Vinař et al., vďaka čomu je náš algoritmus oveľa vhodnejší na hľadanie najlepšej histórie. Detaily nášho algoritmu si ukážeme v podkapitole 3.7.

Súčasťou našej práce je aj implementácia tohto algoritmu. Naprogramovali sme teda kompletný algoritmus na rekonštrukciu jednej histórie, všetky metódy na skórovanie udalostí, vybudovanie stromov lokusov na základe histórie a zarovnaní atómov (pojem atóm vysvetľujeme v nasledujúcej podkapitole). Implementácia umožňuje meniť spôsob skórovania udalostí a zapínať rôzne módy behu programu.

Z vecí, ktoré spomíname, sme neprogramovali algoritmus, ktorý rozdelí sekvenciu

na atómy, atómy do skupín a nájde zarovnania atómov v skupine. Ten už naprogramovala Brejová et al. [BBV11]. Funkciu na počítanie pravdepodobnosti naprogramoval Anderle [And14], použil pri tom rovnaký pravdepodobnostný model, aký je popísaný v kapitole 2.

3.1 Atomizácia sekvencie

Prvým krokom algoritmu je predspracovanie sekvencie na vstupe a jej rozdelenie na tzv. atómy. Cieľom tohto kroku je určiť, ktoré úseky sekvencie si zodpovedajú, čiže vznikli so spoločného predka. „Zodpovedanie si“ je relácia ekvivalencie.

Atóm je vo všeobecnosti úsek sekvencie, pričom atómy sú rozdelené do skupín a spĺňajú nasledovné vlastnosti.

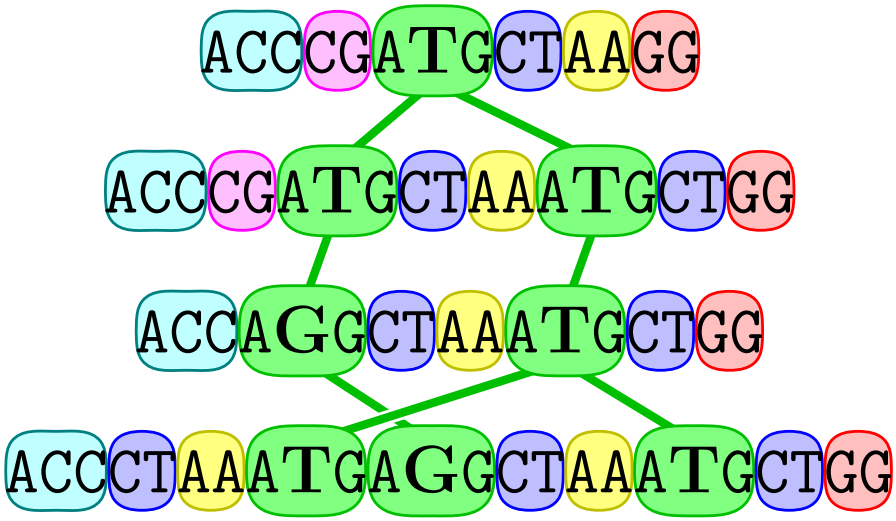
- Žiadne dva atómy sa neprekrývajú.
- Každé dva atómy v jednej skupine si zodpovedajú.
- Keď si nejaké dva úseky sekvencie zodpovedajú, sú podúsekmi atómov v rovnakej skupine.
- Žiadna udalosť nerozdelila žiaden atóm na dve časti. Tým pádom všetky hranice duplikovaných a deletovaných úsekov sú na hraniciach atómov. (Táto podmienka vysvetľuje pôvod slova atóm.)
- Počet skupín je čo najmenší možný.

Príklad rozdelenia sekvencie na atómy môžeme vidieť na obrázku 3.1. Obrázok slúži len na ilustráciu, v praxi je dĺžka sekvencie aj atómov oveľa dlhšia.

Problémom rozdelenia sekvencie na atómy sa nebudeme v tejto práci ďalej venovať, pretože sa mu už venovali tieto práce: Brejová et al. [BBV11] a Višnovská et al. [VVB13]. My budeme predpokladať, že na vstupe máme zoznam atómov v poradí, v akom sa nachádzajú v sekvencii a pre každú skupinu máme zarovnania atómov v nej.

Všimnime si, že ak by sme vynechali poslednú podmienku, tak by vyhovujúce rozdelenie na atómy bolo aj také, že každé písmeno by bol jeden atóm a každý lokus by bola skupina atómov. Atómy a lokusy totiž veľmi úzko súvisia, ako si ukážeme v podkapitole 3.3.

Avšak podmienkou o najmenšom počte získame v praxi dve výhody. Ako prvé, počet atómov bude oveľa menší, ako dĺžka sekvencie. Zatiaľ čo sekvencie, s ktorými pracujeme, sú dlhé stotisíc až pol milióna báz, počty atómov budú v desiatkach až stovkách. Pokiaľ



Obr. 3.1: Sekvencie v histórii rozdelené na atómy. Znázornený je evolučný strom pre zelenú skupinu.

budeme mať napríklad algoritmus, ktorého časová zložitosť bude kubická v závislosti od počtu atómov, tak si tento rozdiel určite všimneme.

Druhá výhoda je, že vďaka vyššej dĺžke atómov dokážeme oveľa lepšie rozhodnúť, či si dané dva atómy zodpovedajú. Zatiaľ čo pri jednopísmenových lokusoch na základe podobnosti toho nevieme veľa určiť, dlhé sekvencie, ktoré vznikli zo spoločného predka sa veľmi podobajú. Podobnosť vieme vyjadriť napríklad ako skóre zarovnania.

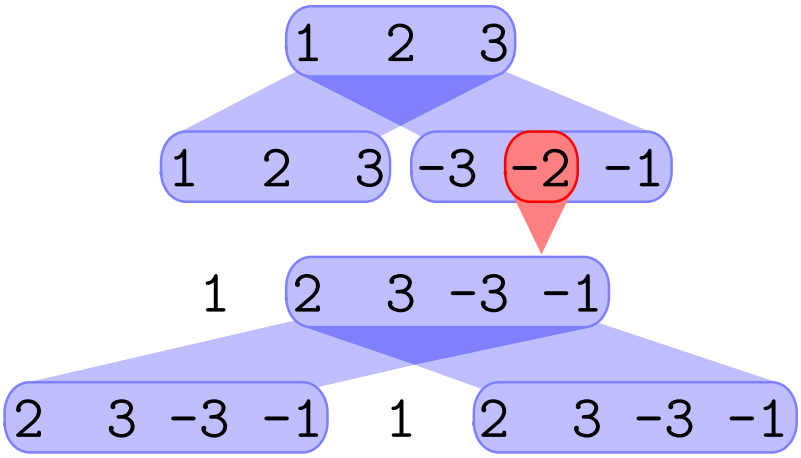
Pre ilustráciu, za 0.04 časovej jednotky, čo je bežný čas s ktorým pracujeme, sa priemerne zmení približne 3.8952% znakov v sekvencii, čo je vypočítané vzorcom 4.4. Je veľmi nepravdepodobné, že by nezávisle vznikli dve takmer identické sekvencie dlhé stovky až desaťtisíce znakov. Pri rozdeľovaní sekvencie na atómy sa nám môže občas stať, že niektoré atómy budú príliš krátke na to, aby sme vedeli rozhodnúť, či patria do nejakej skupiny. S takýmito krátkymi nezaradenými atómami sa vysporiadame jednoducho tak, že ich zmažeme zo sekvencie.

Keď teda budeme vedieť, ktorá časť sekvencie zodpovedá ktorému atómu, a budeme vedieť, ktorý atóm patrí do ktorej skupiny, očíslujeme si skupiny prirodzenými číslami 1 až počet skupín. Číslo skupiny nazveme aj typ atómu.

Následne v sekvencii všetky výskyty atómu z i-tej skupiny nahradíme číslom i alebo $-i$ tak, aby navzájom inverzné atómy mali opačné znamienko. Totiž, ak sú dva atómy navzájom inverzné, chceme ich mať v jednej skupine, pretože predpokladáme, že pochádzajú z toho istého predka.

Takto dostaneme postupnosť celých čísel a našou úlohou teraz bude hľadať históriu takejto postupnosti. Príklad takejto postupnosti a jej história je na obrázku 3.2.

Všimnime si, že v tejto postupnosti už neuvažujeme o mutáciách a zaoberáme sa len postupnosťou udalostí.



Obr. 3.2: Príklad histórie sekvencie atómov. Nastala inverzná duplikácia, delécia a duplikácia.

3.2 Neodhaliteľnosť niektorých udalostí

V našej práci sa musíme vysporiadať s problémom, že v sekvencii môže nastať udalosť, ktorú z princípu nie je možné odhaliť. Tohto charakteru je napríklad veľká časť delécií. Ak by pôvodná sekvencia bola `1 2 3` a v tejto sekvencii by nastala delécia atómu v strede, dostali by sme sekvenciu `1 3`. Z nášho pohľadu by to však bola obyčajná sekvencia, v ktorej by sa nenachádzali žiadne zodpovedajúce si úseky, takže by sme ju považovali za jeden atóm a prehlásili by sme, že žiadna udalosť nenastala. Nič v súčasnej sekvencii totiž nenasvedčuje, že sa nejaká delécia stala, prípadne kedy a kde nastala. Podobne aj delécia na obrázku 1.2 je neodhaliteľná.

Deléciu zvyčajne dokážeme odhaliť len v prípade, keď vymazaná sekvencia bola podsekvenciou nejakej duplikovanej sekvencie. Teda napríklad ak máme na začiatku sekvenciu `1 2 3`, nastane duplikácia s inverziou na `1 2 3 -3 -2 -1` a následne nastane delécia atómu `-2`, dostaneme sekvenciu `1 2 3 -3 -1`. Pri pohľade na túto sekvenciu by sme mali usúdiť, že spomínaná delécia nastala. Je totiž oveľa menej pravdepodobné, že by miesto toho nastali dve duplikácie s inverziou, ktoré by sa ešte trafili tak pekne vedľa seba.

Ani vtedy však nevieme určiť presný čas delécie. Vidíme, že v sekvencii niečo chýba, ale nevieme určiť, kedy táto udalosť nastala. Preto budeme predpokladať, že delécia nastala priamo po tej duplikácii, z ktorej sme zistili jej existenciu. Pri rekonštrukcii histórie tieto dve udalosti združíme do jednej, ktorú voláme duplikačná udalosť.

3.3 Evolučné stromy pre lokusy a atómy

Z rozdelenia sekvencie na atómy vieme odvodiť rozdelenie písmen do lokusov. Totiž dve sekvencie si zodpovedajú práve vtedy, keď sú zarovnané pod sebou v nejakej skupine atómov. Toto platí aj pre sekvencie dĺžky jedna. Každý stĺpec v zarovnaní atómov každej skupiny teda tvorí jeden lokus.

Keď chceme zistiť dĺžku pôvodnej sekvencie, ktorú potrebujeme pri výpočte pravdepodobnosti histórie, spočítame počet lokusov. Čiže sčítame dĺžku (počet stĺpcov) zarovnania pre všetky skupiny.

Z praktických dôvodov (vyššia rýchlosť a nižšie pamäťové nároky), nebudeme budovať strom pre každý lokus zvlášť, ale vystačíme si s evolučnými stromami pre atómy. Spôsob, akým si v algoritme reprezentujeme histórie je prispôsobený na to, aby sa dali stromy atómov budovať ľahko. Každá udalosť má okrem základných informácií aj zoznam atómov, predstavujúci tvar sekvencie po tejto udalosti a pre každý z týchto atómov si pamätáme poradie jeho predka v predošlej sekvencii. V našej reprezentácii histórií si tiež pamätáme, z akých atómov sa skladá počiatočná sekvencia. Táto reprezentácia je veľmi podobná formátu výstupu, ktorý je popísaný v podkapitole 3.9.

Pre každú skupinu atómov teda budeme mať jeden evolučný strom. Po každej udalosti poznáme aktuálnu sekvenciu atómov a pre každý z týchto atómov budeme mať jeden vrchol v evolučnom strome. Synovia tohoto vrcholu budú atómy, ktoré z neho vznikli. Vrcholy na najnižšej úrovni stromu budú poznať súčasné sekvencie prislúchajúcich atómov.

Na takto vybudovaných stromoch už môžeme spustiť Felsensteinov algoritmus, ktorý spočíta vierohodnosť každého lokusového stromu. Táto vierohodnosť je dôležitá pri výpočte pravdepodobnosti histórie v podkapitole 2.4.

3.4 Zrýchlenie výpočtu vierohodnosti

Časová zložitosť Felsensteinovho algoritmu je $O(\ell\,\Sigma^3\,v)$, kde ℓ je dĺžka sekvencie, Σ je veľkosť abecedy a v je počet vrcholov evolučného stromu. Pri praktických testoch sa tento algoritmus ukázal byť príliš pomalý a zároveň výpočet pravdepodobnosti histórie bol najpomalšou časťou celého algoritmu. Preto sme sa rozhodli zrýchliť ho.

Prvou z optimalizácií bolo zjednodušenie evolučného stromu, kde sme odstránili vrcholy, ktoré mali práve jedného syna a mali otca. Syna a otca tohto vrcholu sme prepojili novou hranou, ktorej dĺžka bola súčtom dĺžok pôvodných hrán. Takéto vrcholy vznikajú z atómov, s ktorými sa nič neudialo počas udalosti. Vďaka tejto optimalizácii sa znížilo v, ktoré vystupuje v časovej zložitosti Felsensteinovho algoritmu.

Druhá optimalizácia využíva, že vstup má špecifické vlastnosti. Vďaka tomu, že mutácie sú relatívne zriedkavé, veľmi často sa stane, že všetky listy stromov lokusov majú rovnaké písmeno. Presnejšie, všetky podstromy lokusových stromov, v ktorých nenastala mutácia, majú v listoch rovnaké písmená. Navyše všetky stromy lokusov, ktoré sú súčasťou jednej skupiny atómov, sú navzájom izomorfné. Dva izomorfné (aj s ohľadom na dĺžky hrán) zakorenené stromy s rovnakými písmenami v listoch majú rovnakú vierohodnosť.

Algoritmus na výpočet vierohodnosti stromov sme vylepšili tak, aby pre každý vrchol v amortizovanom konštantnom čase zistil, či má všetky písmenká v listoch rovnaké. Vďaka technike memoizácie sme zabezpečili, aby takéto vrcholy nepočítali svoju vierohodnosť, ak už bola predtým vypočítaná v rovnakom strome s rovnakými písmenami v listoch. Časová zložitosť algoritmu sa tým znížila na $O(m\,\Sigma^3\,v + \ell\,\Sigma)$, kde m je počet celých stromov lokusov, ktoré majú rôznorodé listy. V dátach s ktorými pracujeme $m \ll \ell$, takže táto optimalizácia sa zdá byť veľmi výhodná.

Tretí spôsob zrýchlenia algoritmu úplne vynechá počítanie vierohodnosti stromov. Pravdepodobnosť histórie by sme mohli definovať jednoducho $P(H|\ell_0)$, ale uznali by sme len tie histórie, ktoré vygenerujú správnu sekvenciu atómov. V podstate by sme z počítania pravdepodobnosti vynechali len mutácie. Zatiaľ čo predošlé dva spôsoby optimalizácie neovplyvňovali výsledok programu, o tejto to povedať nemôžeme. Dôvod, prečo nám to nevadí, je ten, že sa domnievame, že nás tento spôsob hodnotenia stále povedie k správnej histórii. Teda že skutočná história bude mať zo všetkých týchto histórii stále najvyššiu pravdepodobnosť.

V 5. kapitole ukážeme, že v praxi tieto optimalizácie, čo sa týka času, výrazne pomôžu. Ukážeme si tiež dopad tretej optimalizácie na nájdenú históriu. Teraz sa poďme pozrieť na samotné rekonštruovanie histórie.

3.5 Predchádzajúce prístupy k rekonštrukcii histórie

Ako sme spomínali na začiatku kapitoly, tejto téme sa už Vinař et al. [VBSS10] a Kravec [Kra11]. Algoritmus podľa Kravca funguje nasledovne.

1. Vstup: súčasná sekvencia atómov.
2. Opakujeme n krát:
   (a) Kým nemáme celú históriu, teda kým sa v sekvencii opakuje nejaký typ atómu, opakujeme:
      i. Navrhneme všetky duplikačné udalosti konzistentné s aktuálnou sekvenciou.
      ii. Náhodne vyberieme jednu udalosť.
      iii. Uložíme udalosť do histórie.
      iv. Vyrobíme sekvenciu pred touto udalosťou a nastavíme ju ako aktuálnu.
   (b) Vyrátame vierohodnosť novej histórie.
   (c) Porovnáme so starou históriou.
   (d) Rozhodneme sa, či prijmeme novú históriu a ak áno, uložíme ju ako starú. (Pokiaľ je história lepšia ako stará prijmeme ju, pokiaľ nie je lepšia tak ju prijmeme s určitou pravdepodobnosťou)

Krok 2d súvisí so spomínaným MCMC algoritmom, podľa ktorého pri výbere udalostí v kroku 2(a)ii zvýhodňujeme udalosti, ktoré sa vyskytovali v predošlej histórii. Podstatný rozdiel v algoritmoch Vinaňa et al. a Kravca bol v krokoch 2(a)i a 2(a)ii.

Vinař et. al vyberali spomedzi všetkých možných duplikačných udalostí. Následne každej z týchto udalostí priradili skóre, ktoré rozhodovalo o pravdepodobnosti ich výberu. Problém tohto prístupu bola časová zložitosť, keďže výber jednej duplikácie trval až $O(n^5)$, kde n je dĺžka sekvencie v atómoch.

Kravec sa tento prístup snažil zlepšiť a navrhol algoritmus na navrhovanie a výber udalosti s časovou zložitosťou $O(n^2)$. Navyše tento algoritmus dovoľoval, aby po duplikácii nasledoval ľubovoľný počet delécií. Nevýhodou tohto algoritmu však bola príliš jednoduchá skórovacia funkcia, ktorá závisela len od dĺžky duplikovanej časti, počtu a dĺžok delécií a toho, či sa udalosť nachádzala v predošlej histórii.

3.6 Vzorkovanie duplikačnej udalosti pomocou DP

V tejto podkapitole si trocha priblížime algoritmus, ktorý používa Kravec [Kra11] vo svojej práci na vzorkovanie jednej duplikačnej udalosti a ktorý sme v našej práci použili

na podobné účely. Najskôr pre jednoduchosť uvažujme situáciu bez inverzných duplikácií a delécií, čiže sa zaoberajme len obyčajnými duplikáciami. Nech počet atómov v celej sekvencii je n a typy týchto atómov sú postupne $a_1, a_2, \ldots, a_n$ (inverzné atómy majú typ vynásobený -1). Chceme, aby skóre duplikácie dĺžky ℓ bolo $2^{\ell-1}$.

Vytvoríme si orientovaný ohodnotený graf, ktorý ma dva význačné vrcholy B a E a maticu vrcholov S rozmerov $n \times n$. Vrchol $S_{i,j}$ označíme ako aktívny práve vtedy, keď $i \neq j$ a $a_i = a_j$. Z B do každého aktívneho vrchola vedie hrana s cenou 1 a z každého aktívneho vrchola vedie hrana do E s cenou 1. Keď $S_{i,j}$ aj $S_{i+1,j+1}$ sú oba aktívne vrcholy, tak z $S_{i,j}$ do $S_{i+1,j+1}$ vedie hrana s cenou 2.

Cena cesty nech je súčin cien všetkých hrán na nej. Potom každá cesta z B do E vedie cez postupnosť vrcholov $S_{i,j}, S_{i+1,j+1}, \ldots, S_{i+\ell-1,j+\ell-1}$, pre nejaké i, j, ℓ. Zároveň platí, že $a_i, \ldots, a_i + \ell - 1 = a_j, \ldots a_j + \ell - 1$, čo znamená, že tieto dva úseky postupnosti mohli vzniknúť duplikáciou. Navyše cena cesty je zhodná so skóre duplikácie. Prirodzene teda takejto ceste priradíme duplikáciu pozícií $i, \ldots, i + \ell - 1$ na pozície $j, \ldots, j + \ell - 1$.

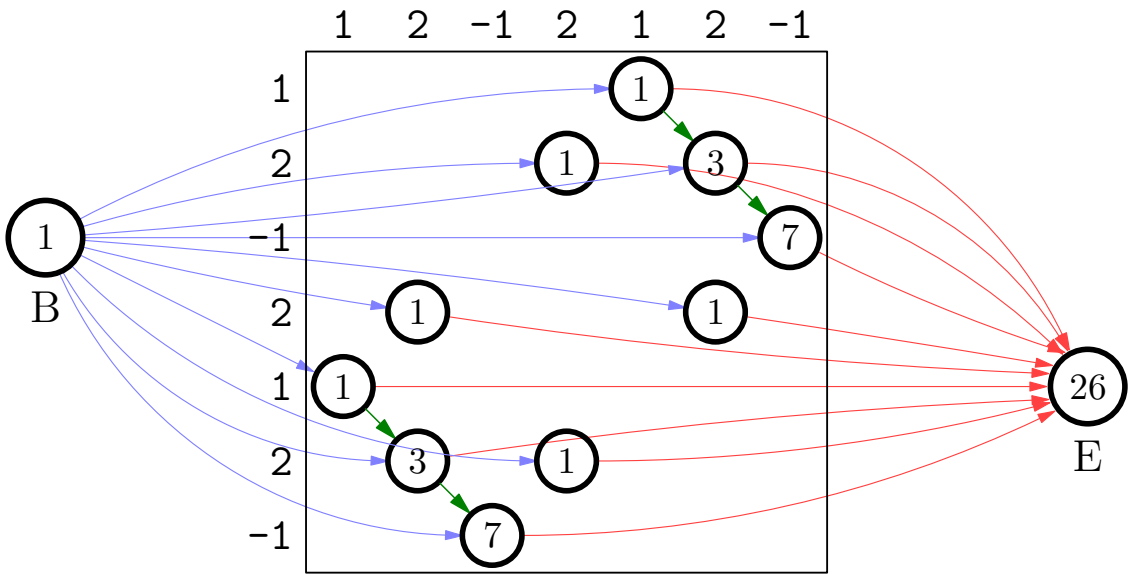
Dynamickým programovaním si môžeme pre každý vrchol v grafe spočítať súčet cien všetkých ciest, ktoré v ňom končia. Tiež si vieme nájsť všetky hrany, ktoré do neho vedú. Vďaka týmto údajom dokážeme vytvoriť algoritmus, ktorý spätným stochastickým prechodom vyberie jednu konkrétnu duplikáciu, pričom každá má pravdepodobnosť výberu úmernú jej skóre. Tomuto algoritmu sa môže stať, že nájde prekrývajúce sa úseky, teda $i \leq j < i + \ell$ alebo $j \leq i < j + \ell$, takže takéto duplikácie zahadzujeme. Príklad grafu môžeme vidieť na obrázku 3.3.

Keď uvažujeme aj duplikácie s inverziou, pridáme do grafu druhú maticu vrcholov S', ktorá namiesto porovnávania a_i a a_j bude porovnávať a_i a $-a_{n-i+1}$. Delécie do algoritmu zakomponujeme tak, že pridáme ďalšie štyri matice, dve budú reprezentovať delécie po obyčajnej duplikácii, dve po inverznej. Detailom sa nebudeme venovať, podrobne tento algoritmus popísal Kravec [Kra11].

V našej implementácii algoritmu používame trocha odlišné ceny hrán, pretože sme tak dosahovali lepšie výsledky. Hrana medzi dvoma aktívnymi vrcholmi má cenu len 1.7 namiesto 2.

3.7 Náš algoritmus na rekonštrukciu histórie

V našom prístupe sme skombinovali algoritmus z predošlej podkapitoly 3.6 s algoritmom, ktorý používal Vinař et al. [VBSS10] tak, aby sme dostali rýchly algoritmus, ktorý má zároveň čo najlepší spôsob výberu udalosti. Naďalej budujeme históriu od konca, od udalostí, ktoré sú najbližšie k súčasnosti, až po tie najvzdialenejšie.



Obr. 3.3: Ukážka grafu pre postupnosť `1 2 -1 2 1 2 -1`. Zobrazené sú len vrcholy s nenulovým stupňom. Modré a červené hrany majú cenu 1. Zelené hrany majú cenu 2. Čísla vo vrcholoch udávajú súčet cien ciest, ktoré vo vrchole končia.

Každú udalosť nájdeme tak, že algoritmom navrhnutým podľa podkapitoly 3.6 necháme vygenerovať $O(n)$ návrhov na udalosť, pre tieto návrhy vypočítame skóre podobným spôsobom ako Vinař et al. a nakoniec vyberieme udalosť náhodne s pravdepodobnosťami priamo úmernými tomuto skóre. Spôsob skórovania udalosti popíšeme v kapitole 4. V algoritme sme tiež nahradili metódu MCMC jednoduchším prístupom.

Keď L je čas, ktorý trvá vypočítanie skóre pre jednu duplikačnú udalosť, tak celková časová zložitosť navrhnutia jednej udalosti je $O(n^3 + n\,L)$, čo by sa dalo zrýchliť na $O(n^2 + n\,L)$, ale v praxi by sa toto zrýchlenie neprejavilo (lebo multiplikatívny člen pri n^3 je pri bežných vstupoch veľmi malý v porovnaní s členom pri n^2). Kompletný výsledný algoritmus teda vyzerá nasledovne:

1. Vstup: súčasná sekvencia atómov a ich zarovnania
2. Opakujeme n krát:
   (a) Do S priradíme sekvenciu na vstupe v podobe postupnosti atómov.
   (b) Kým sa nejaký typ atómu vyskytuje v sekvencii viackrát, opakujeme:
      i. Navrhneme $O(n)$ kandidátov na udalosti E, algoritmom z podkapitoly 3.6.
      ii. Pre každú udalosť $e \in E$ vypočítame skóre $s(e)$.

iii. Vyberieme náhodnú udalosť tak, že udalosť e má pravdepodobnosť výberu
$$\frac{s(e)}{\sum_{e' \in E} s(e')}.$$
iv. Zrekonštruujeme sekvenciu pred touto udalosťou, takzvanú ancestrálnu sekvenciu, a uložíme ju do S.
v. Pridáme udalosť do zoznamu udalostí E.

(c) História bude zoznam udalostí v E.

(d) Vybudujeme stromy lokusov.

(e) Spočítame pravdepodobnosť novej histórie.

(f) Ak je nová história lepšia ako predošlá, poznačíme si všetky jej udalosti.

3. Vypíšeme najlepšiu zo všetkých histórií

Viac o formáte vstupu a výstupu si povieme v podkapitole 3.9. Údaje z predchádzajúcich histórií, ktoré si poznačíme v kroku 2f využijeme pri skórovaní popísanom v podkapitole 4.5.

3.8 Určovanie časov udalostí

Pre dokončenie myšlienky algoritmu nám ostáva už len povedať, ako budeme jednotlivým udalostiam priraďovať časy, kedy nastali. Stanovenú máme len dobu medzi prvou a poslednou udalosťou.

Jedna možnosť je, že sa týmto problémom teraz nebudeme veľmi trápiť a jednoducho rozmiestnime udalosti v pravidelných rozostupoch medzi počiatočný čas a súčasnosť. Domnievame sa totiž, že čas má na pravdepodobnosť histórie menší vplyv ako napríklad správna množina alebo správne poradie udalostí. Či to je naozaj tak, overíme v kapitole 5.

Druhá možnosť je využiť zarovnanie atómov. Zjednodušene povedané, čím viac sa na seba DNA sekvencie atómov podobajú, tým menej dávno nastala duplikácia, ktorá ich rozdelila. Podrobnosti sú rozpísané v podkapitole 4.2. Podľa vzorca 4.5 určíme vzdialenosti medzi udalosťami a tie potom prenásobíme vhodným koeficientom, aby sa dokopy nasčítali na stanovený čas. Bohužiaľ tento prístup je v praxi pomalý a teda ho budeme používať iba v prípade, že je zapnutá skórovacia metóda z tejto podkapitoly.

3.9 Formát vstupu a výstupu

Náš algoritmus bude mať na vstupe zoznam atómov nachádzajúcich sa v skúmanej DNA sekvencii. Tento zoznam je v textovom formáte, každý atóm na jednom riadku, zoradené zľava doprava, ako sa nachádzajú v sekvencii. Popis atómu sa skladá zo šiestich údajov: názov organizmu, názov atómu, číslo skupiny, orientácia (normálna je 1, inverzná -1), pozícia začiatku a pozícia konca v pôvodnej sekvencii (používame polouzavreté intervaly). Príklad takéhoto súboru môžeme vidieť na obrázku 3.4.

```
clovek c1 2  1      0  27174
clovek c2 3  1  27174  30447
clovek c3 3 -1  30447  33720
clovek c4 1 -1  33720  50030
clovek c5 1  1  50030  66340
clovek c6 2  1  66340  93514
clovek c7 3  1  93514  96787
clovek c8 3 -1  96787 100060
clovek c9 1 -1 100060 119819
```

Obr. 3.4: Ukážka súboru so zoznamom atómov. Zodpovedá poslednej sekvencii z obrázku 3.2.

Okrem toho, pre každú skupinu dostaneme jeden súbor so zarovnaniami atómov v tejto skupine. Jeden z týchto súborov môže vyzerať tak, ako na obrázku 3.5.

```
>c4
AAGATGGTTATAC-GTGACT..ďalších 16280 znakov..TGAACGAGTT
>c5
AAGCTGGTTATGCCGTGACT..ďalších 16280 znakov..TGCACGAGTT
>c9
AAGATGGTTATAC-GTGACT..ďalších 16280 znakov..TGAACGAGAT
```

Obr. 3.5: Príklad súboru so zarovnaniami 1. skupiny atómov.

Jedným výstupom programu sú štatistické údaje z priebehu programu, ktoré sú dôležité pre testy v kapitole 5. Druhým výstupom je najlepšia nájdená história vo formáte, aký používame na prácu s históriami a ich ukladanie.

Výstupný formát histórie je zoznam riadkov. Prvý riadok popisuje, ako vyzerala sekvencia na začiatku, každý ďalší riadok popisuje jednu udalosť. V riadku je postupne čas a typ udalosti, zoznam atómov v sekvencii po udalosti a pozície predkov týchto

atómov v sekvencii pred udalosťou. Príklad výstupného súboru môžeme vidieť na obrázku 3.6.

```
0 root 1 3 -2 4 # -1 -1 -1 -1
0.01 dup 1 3 -2 4 3 -2 # 0 1 2 3 1 2
0.02 dupi -3 -4 2 -3 1 3 -2 4 3 -2 # 4 3 2 1 0 1 2 3 4 5
0.03 del -3 -4 2 -3 1 3 4 3 -2 # 0 1 2 3 4 5 7 8 9
0.04 leaf -3 -4 2 -3 1 3 4 3 -2 # 0 1 2 3 4 5 6 7 8
```

Obr. 3.6: V tejto histórii nastali tri udalosti – duplikácia, duplikácia s inverziou a delécia. Posledná udalosť je špeciálna koncová udalosť a vyjadruje súčasný stav.

Kapitola 4

Skórovanie duplikačných udalostí

Cieľom tejto kapitoly je navrhnúť spôsob skórovania udalostí, presnejšie zostrojiť funkciu, ktorá výslednej sekvencii a duplikačnej udalosti priradí reálne číslo – skóre udalosti.

Túto funkciu používame v algoritme spomínanom v podkapitole 3.7. Slúži na to, aby sme vedeli čo najlepšie zvoliť udalosť spomedzi množiny kandidátov E. Pokiaľ skóre udalosti e označíme $s(e)$, tak pravdepodobnosť výberu tejto udalosti je

$$\frac{s(e)}{\sum_{e' \in E} s(e')} \tag{4.1}$$

Navrhneme niekoľko rôznych metód, ktoré rôznymi spôsobmi určujú, ktorá duplikačná udalosť pravdepodobne nastala. Každá z týchto metód priradí udalosti čiastkové skóre a výsledné skóre vznikne funkciou týchto čiastkových skóre. Okrem toho penalizujeme každú deléciu empirickou konštantou e^{-1}.

Nech s_i je čiastkové skóre i-tej metódy a v_i je kladná váha, ktorú tejto metóde priradíme. Čím vyššia je váha, tým väčšmi má toto čiastkové skóre vplyv na výsledné skóre udalosti s, ktoré vypočítame nasledovným vzťahom (d je počet delécií).

$$s = e^{-d+\sum_i^6 s_i \cdot v_i} \tag{4.2}$$

Umožníme v programe jednoducho meniť váhy v_i pomocou konfiguračného súboru, aby sme mohli potom v praktických testoch skúmať vplyv jednotlivých metód na správnu rekonštrukciu histórie.

V nasledujúcich podkapitolách si popíšeme jednotlivé metódy skórovania, resp. ako vypočítať s_1 až s_6.

4.1 Dĺžka duplikácie

Vo väčšine prípadov je najlepšia história danej sekvencie zároveň aj najkratšia história tejto sekvencie, čo sa týka počtu udalostí, čo vyplýva z toho, ako je postavený pravdepodobnostný model. Preto budeme uprednostňovať udalosti, ktoré vedú ku kratším históriám, tzn. udalosti, ktoré zahrňujú veľké množstvo atómov. Ak duplikačná udalosť pozostáva z d atómov, tak $s_1 = \log(d)$.

Problém je tiež aj to, že ak by sme nepenalizovali krátke udalosti, mohla by byť príliš veľká pravdepodobnosť, že história bude len postupné duplikovanie atómov po jednom, čo má väčšinou veľmi ďaleko od správnej histórie.

4.2 Podobnosť sekvencií

V momente ako nastane v DNA duplikácia, duplikované úseky sú rovnaké. Postupne ako plynie čas a dejú sa mutácie, začínajú sa dva úseky, ktoré vznikli z pôvodného predka, čoraz viacej odlišovať. Preto očakávame, že atómy v dvojici, ktorá vznikla z nedávnej duplikácie, budú na seba podobnejšie ako tie, ktoré vznikli dávnejšie.

Potrebujeme preto vyjadriť vzťah medzi počtom odlišností v dvojici sekvencií a množstvom času, ktorý uplynul od spoločného predka. Podľa pravdepodobnostného modelu vyjadríme pravdepodobnosť, že sa písmeno na jednej pozícii za čas t nezmení takto:

$$P(a|a,t) = \frac{1}{4}(1 + 3e^{-4/3t}) \tag{4.3}$$

Pre očakávaný počet p nezmenených písmen v celej sekvencii dĺžky ℓ platí

$$E(p) = \ell \cdot \frac{1}{4}(1 + 3e^{-4/3t}) \tag{4.4}$$

$$t = -\frac{3}{4}\log(\frac{4E(p)}{3\ell} - \frac{1}{3}) \tag{4.5}$$

Keď teda budeme chcieť odhadnúť, ako dávno mali dva atómy spoločného predka, spočítame na koľkých miestach sa zhodujú. Na základe predošlého vzťahu vypočítame čas t, ktorý vydelíme dvoma, pretože obe sekvencie sa vyvíjali nezávisle odkedy sa rozdelili. Pri skórovaní duplikácie by sme chceli vyššie skóre dať udalostiam, ktoré majú vážený priemer časov (váhy sú dĺžky atómov) najkratší spomedzi všetkých duplikovaných atómov. Vysoká disperzia týchto časov napovedá, že sme nevybrali správnu duplikáciu, čiže ju tiež započítame do skóre:

$$s_2 = \frac{\mu + \sigma}{t_h}, \tag{4.6}$$

kde μ je priemer časov, σ je ich disprezia a t_h je celková doba histórie.

Aby sme mohli použiť túto metódu skórovania, potrebujeme vedieť určiť aj podobnosť atómov, ktorých sekvencie nemáme na vstupe. Napríklad, ak pôvodná sekvencia na vstupe bola `1 2 3 1 2 -1 -2 3` a rozhodneme sa, že posledná udalosť bola inverzná duplikácia štvrtého a piateho atómu, zrekonštruujeme postupnosť atómov pred duplikáciou na `1 2 3 1 2 3`. Nevieme však povedať, ako vyzerali pred duplikáciou ich DNA sekvencie. Preto každej pozícii v sekvencii priradíme štvorrozmerný vektor, $v \in \langle 0, 1\rangle^4$, ktorého súradnice budú postupne vyjadrovať pravdepodobnosti, že na danej pozícii je príslušné písmeno z {`A`,`C`,`G`,`T`}. Predka dvoch vektorov vypočítame ako ich aritmetický priemer. „Rovnakosť“ dvoch vektorov v_1 a v_2 označme $p(v_1, v_2)$ a definujeme ju ako kosínus uhla medzi týmito vektormi, čiže

$$p(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|} \tag{4.7}$$

Nevýhoda tejto metódy je to, že časová zložitosť je závislá nielen od počtu atómov, ako je to pri ostatných metódach, ale aj od počtu báz samotnej sekvencie.

4.3 Čiastočná duplikácia

Predstavme si, že nastane duplikácia úseku $a_1 a_2 \dots a_l$. Potom sa vo výslednej sekvencii bude vyskytovať tento úsek dvakrát, a teda v navrhnutých duplikáciách môžeme mať aj všetky možné časti tejto duplikácie $a_j \dots a_k$ pre ľubovoľné $1 \leq j \leq k \leq l$. Aby sme sa tomuto problému vyhli, budeme penalizovať kratšie udalosti.

Ak sa skúmaná udalosť dá predĺžiť len doľava alebo len doprava, s_3 bude -1, ak do oboch strán, s_3 bude -2, inak bude mať s_3 hodnotu 0.

4.4 Zníženie rôznorodosti atómov

V počiatočnej sekvencii je n atómov rôznych typov. Každý typ atómu susedí s najviac jedným typom atómu na ľavej strane a najviac s jedným na pravej. Inverzné atómy budú mať opačne ľavú a pravú stranu.

Pri duplikácii postupnosti atómov $a_1 \dots a_l$ medzi atómy b, c vzniknú dve dvojice typov atómov (b, a_1) a (a_l, c). S veľkou pravdepodobnosťou sa v sekvencii zatiaľ nevyskytovali dvojice takýchto typov atómov. Možných dvojíc typov je $O(n^2)$, a po d udalostiach sa v sekvencii vyskytuje najviac $2n+2d$ rôznych dvojíc typov. Pri hodnotení udalosti si spočítame, koľko párov susedných atómov by touto udalosťou ubudlo a nastavíme s_4 na tento počet.

Môže sa nám zdať, že s_4 bude úzko súvisieť s s_3, ale nie je to to isté. V ďalšej kapitole môžeme tieto metódy porovnať a zistiť, ktorá dáva lepšie výsledky.

Taktiež v počiatočnej sekvencii sa každý typ atómu vyskytuje buď len s kladným alebo len so záporným znamienkom. Pri inverznej duplikácii sa väčšinou zvýši počet typov, ktoré sa vyskytujú s oboma znamienkami. Čím menej takýchto typov v sekvencii bolo, tým väčšia je očakávaná hodnota tohto nárastu. Ak nám inverzná duplikácia zníži počet obojznamienkových typov o a a ešte tam ostane b takýchto typov, tak ku s_4 pripočítame $\log(1 + \frac{a}{1+b}) - 0.5$, čo sme určili na základe empirického odhadu.

Počet rôznych dvojíc susedných typov plus počet typov atómov, ktoré sa v sekvencii vyskytujú s oboma znamienkami nazveme rôznodorodosť sekvencie. Táto metóda teda vychádza z toho, že rôznorodosť súčasnej sekvencie je oveľa vyššia ako rôznorodosť počiatočnej. Preto uprednostňuje udalosti, ktoré rôznorodosť znížia.

4.5 Výskyt v predošlých históriách

Cieľom tejto metódy je uprednostňovať udalosti, ktoré sa vyskytli v nedávnych históriách. Je to náhrada MCMC algoritmu spomínaného v predošlej kapitole. Metóda funguje nasledovne:

Každej udalosti priradíme jedno reálne číslo, ktoré nazveme obľúbenosť. Keďže potenciálnych udalostí je veľmi veľa, v programe si udržiavame tie udalosti, ktorých obľúbenosť je vyššia ako nejaká pevne zvolená konštanta. Zvyšné udalosti budú mať obľúbenosť 0.

Na začiatku je obľúbenosť všetkých udalostí 0, ale vždy keď sa algoritmu podarí zlepšiť pravdepodobnosť histórie oproti poslednej nájdenej, udalostiam z tejto histórie sa zvýši obľúbenosť o číslo z intervalu $\langle 1, 2\rangle$ podľa toho aká je pravdepodobnosť novej histórie oproti doteraz najlepšej nájdenej. Ostatným udalostiam sa pri tom obľúbenosť trocha zníži. Po každej vygenerovanej histórii sa obľúbenosť každej udalostí prenásobí konštantou 0.9, takže obľúbenosť klesá exponenciálne.

Zabezpečíme, aby sa udalosti, ktoré majú dostatočne vysokú obľúbenosť, určite dostali medzi skórované udalosti, aj keby neboli vybraté algoritmom z podkapitoly 3.6. Túto obľúbenosť zakomponujeme do skórovania tak, že s_5 bude rovné súčtu obľúbeností všetkých udalostí, ktoré vedú k rovnakej ancestrálnej sekvencii.

4.6 Dobrá minulosť

Pri duplikácii nestačí správne uhádnuť, ktoré dva rovnaké úseky sekvencie sú výsledkom tejto duplikácie, ale potrebujeme aj správne určiť, ktorý z týchto úsekov bol v ancestrálnej sekvencii. Napríklad ak uhádneme, že v sekvencii `1 2 3 3 1 2 3 1` nastala duplikácia `1 2 3`, nie je jedno, či ancestrálna sekvencia bola `1 2 3 3 1` alebo `3`

`1 2 3 1`. V tomto prípade je pravdepodobnejšia druhá z nich, pretože tá sa dá vyrobiť zo sekvencie dĺžky 3 jednou duplikáciou. Prvá sekvencia by na to potrebovala najmenej dve udalosti.

Tento problém čiastočne rieši štvrtá skórovacia metóda, ale radi by sme preskúmali aj iné možnosti ako ho vyriešiť. Každej sekvencii by sme chceli priradiť číslo, ktoré vyjadruje, ako dobre sekvencia vyzerá. Následne môžeme hodnotiť pozitívnejšie tie udalosti, ktoré vedú k lepšie vyzerajúcim ancestrálnym sekvenciám.

Číslo určujúce, ako dobre vyzerá sekvencia, bude vychádzať z podkapitoly 3.6 a bude vypočítané na základe hodnoty vo vrchole E, ktorú označíme m. Za normálnych okolností nastavíme $s_6 = \log(m)$. Pre veľmi malé m, keď už sa blížime ku koncu rekonštrukcie, bude s_6 pevne zvolená konštanta. Inak by metóda mohla zbytočne zvyšovať počet udalostí v histórii.

Drobná nevýhoda tejto metódy môže byť to, že jej časová zložitosť, narozdiel od ostatných metód, nezávisí od počtu atómov v sekvencii lineárne, ale až kvadraticky.

Kapitola 5

Testovanie algoritmu, porovnania a výsledky

Jedným z cieľov práce bolo preskúmať účinnosť rôznych možností skórovania udalostí. Pripomeňme si, že celkové skóre udalostí závisí od šiestich čiastkových skóre, ktorým môžeme priradiť rôzne váhy. Budeme sa teda zaoberať rôznymi nastaveniami týchto váh a skúmaním ich vplyvu na účinnosť celého algoritmu.

Vstup	Počet typov atómov	Počet atómov	Dĺžka DNA sek.	Dĺžka histórie
A	22	50	192106	8
B	28	70	221723	10
C	31	103	260552	11
D	42	200	174585	15

Tabuľka 5.1: Parametre vstupov

V tabuľke 5.1 sú popísané parametre vstupov, na ktorých sme program testovali. Vstupy sú zoradené podľa počtu atómov a zároveň subjektívne podľa obtiažnosti.

Budeme používať dva spôsoby počítania pravdepodobnosti histórie. Nazveme ich miera P a miera Q. Miera P bude jednoduchší spôsob a má hodnotu $P(H|\ell_0)$. Oproti miere Q neberie do úvahy mutácie. Mieru Q vypočítame ako $P(H, s|\ell_0)$, čiže $P(s|H, \ell_0) \cdot P(H|\ell_0)$. Spôsob počítania oboch týchto pravdepodobnosti je popísaný v 2. kapitole.

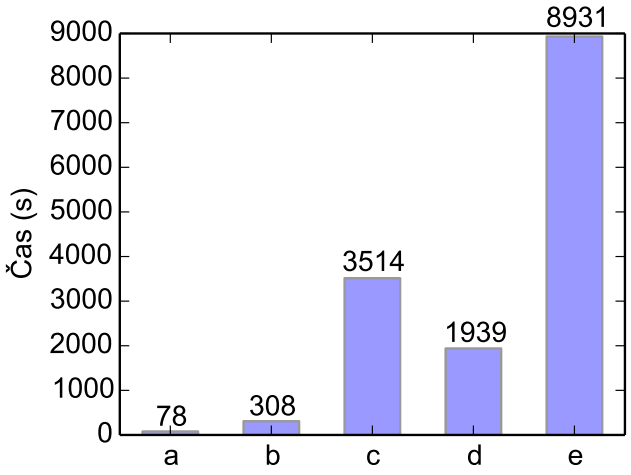
Pôvodná motivácia použitia P bola rýchlosť výpočtu. Neskôr sme objavili aj iné výhody, ale aj nevýhody. Viac o tom, ako dopadli tieto metódy v testoch si povieme v podkapitole 5.3.

5.1 Čas behu programu

Súčasťou porovnania metód skórovania je zistenie, aké rýchle sú jednotlivé metódy. Pokiaľ sa napríklad metóda skórovania ukáže byť príliš pomalá v porovnaní s tým, ako veľmi pomôže pri rekonštrukcii histórie, môžeme sa rozhodnúť túto metódu nepoužiť a radšej stihnúť viac iterácií algoritmu.

Všetky testy boli robené na počítači so 64-bitovým Linuxom 3.2.0-29, s procesorom Intel Xeon CPU taktovanom na 2.27GHz, 8MB CPU cache a 2 GB pamäťou RAM. Kompilovali sme s GCC 4.8.2 so štandardom gnu++11 a optimalizáciou O2.

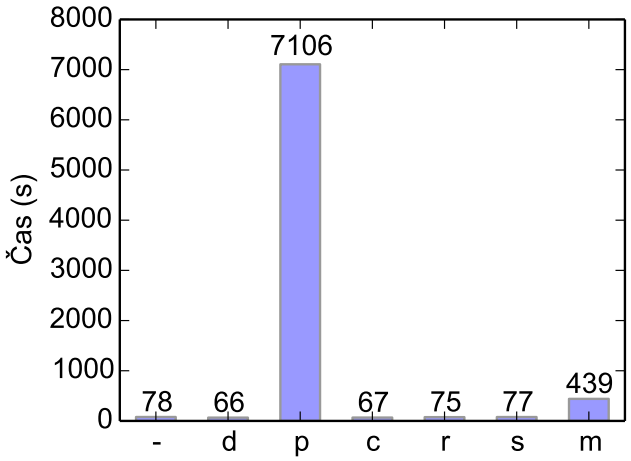
V kapitole 3.4 sme si predstavili zrýchlenie tej časti algoritmu, ktorá počítala pravdepodobnosť histórie, takže si teraz ukážeme, ako veľmi toto zrýchlenie pomohlo. V programe na rekonštrukciu histórie sme vypli všetky metódy skórovania, nastavili sme 1 000 iterácii a spustili sme program na vstupe C. Najprv sme skúšali počítanie pravdepodobnosti ako mieru P. Následne sme štyrikrát skúsili počítanie miery Q so zapnutými a vypnutými optimalizáciami. Ako to dopadlo, je vidieť na obrázku 5.1. Môžeme si všimnúť, že samotné optimalizácie pri výpočte vierohodnosti génových stromov zrýchlili algoritmus skoro 30 násobne, čo považujeme za úspech.



Obr. 5.1: Porovnanie rýchlostí výpočtu pravdepodobnosti histórie.
(a) Miera A
(b) Miera B s oboma optimalizáciami
(c) Miera B, len prvá optimalizácia
(d) Miera B, len druhá optimalizácia
(e) Miera B, žiadna optimalizácia

Rôzne metódy skórovania sme testovali tak, že sme nastavili najrýchlejší spôsob počítania pravdepodobnosti histórie (mieru P) a postupne sme vypínali a zapínali jednotlivé metódy skórovania. Vypnutá metóda mala pri skórovaní váhu 0, zapnutá váhu 1. Opäť sme nechali v programe bežať 1 000 iterácií na vstupe C. Výsledky sú na obrázku 5.2. Spomeňme si, že metódy označené „d“,„c“,„r“ a „s“ mali časovú zložitosť $O(n)$, časová zložitosť metódy „m“ bola $O(n^2)$ a metóda „p“ mala časovú zložitosť $O(n + \ell)$, kde n je počet atómov v sekvencii a ℓ je dĺžka DNA sekvencie.

Zaujímavé je, že keď zapneme nejakú z metód „d“, „c“, „r“ alebo „s“, algoritmus sa zrýchli. Je to preto, že tieto metódy už pri váhe 1 spôsobujú rekonštrukciu kratších histórií, čo sa prejaví na výslednom čase.



Obr. 5.2: Porovnanie rýchlostí metód skórovania
(-) Žiadne
(d) Dĺžka duplikácie
(p) Podobnosť sekvencií
(c) Čiastočné duplikácie
(r) Rôznorodosť atómov
(s) Staré histórie
(m) Dobrá minulosť

5.2 Spôsob testovania a hodnotenia úspešnosti

Algoritmus budeme testovať na simulovaných dátach z toho dôvodu, že keď budeme poznať správnu históriu, budeme môcť určiť, ako bol algoritmus pri jej hľadaní úspešný. Simulované dáta boli vyrobené tak, aby simulovali evolúciu podľa pravdepodobnostného modelu z kapitoly 2. Softvér na výrobu simulovaných dát a spôsob ich výroby, popísali Vinař et al. [VBSS10]. Na rozdiel od nich sme používali dáta len pre jeden organizmus.

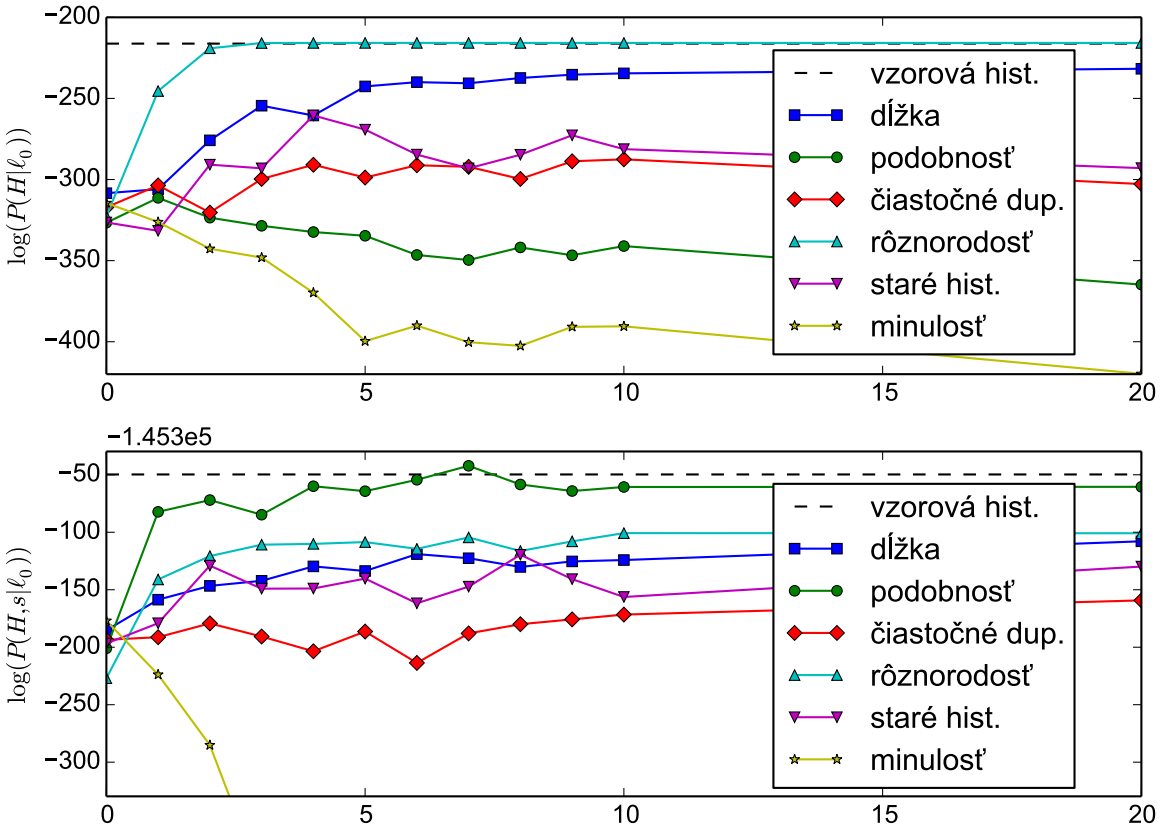
Úspešnosť algoritmu budeme hodnotiť jednoducho tak, že porovnáme pravdepodobnosť najlepšej histórie akú algoritmus našiel s pravdepodobnosťou pôvodnej histórie. Necháme algoritmus postupne vygenerovať 1 000 histórií. Za normálnych okolností by nás zaujímala len najlepšia zo všetkých týchto histórií, ale ľahko by sa mohlo stať, že objektívne horšia skórovacia metóda by mala šťastie a našla by lepšiu históriu ako objektívne lepšia skórovacia metóda. Aby sme znížili takúto chybu merania, rozdelíme 1 000 iterácií na 5 nezávislých blokov po 200 iterácií a celkový výsledok bude priemer z piatich víťazov.

V prípade, že nás nejaký výsledok zaujme, napríklad nejaká metóda nedosiahne očakávané výsledky, môžeme sa následne pozrieť na vygenerovanú najlepšiu históriu (zo všetkých $1,000$ iterácií) a na základe nej môžeme určiť v čom je problém, alebo prečo sú výsledky testovania také, aké sú. Keď pravdepodobnosť histórie nájdenej algoritmom bude blízka tej pôvodnej, skontrolujeme, či algoritmus uspel a našiel ju. Drobné odchýlky pri pravdepodobnosti totiž môžu byť spôsobené tým, že nevieme správne určiť časy udalostí.

5.3 Výsledky testovania

Najprv sa pozrieme na každú metódu skórovania samostatne. Zapnutá bude vždy len jedna zo šiestich metód a budeme skúmať, ako sa mení úspešnosť algoritmu pri zmene váhy. Čím väčšia je váha, tým silnejší je vplyv metódy pri výbere udalosti. Váhy budeme postupne nastavovať na $1, 2, \ldots 10$ a ešte na 20, aby sme overili, či sa niečo výrazne zmení, keď bude váha extrémne vysoká. Celú procedúru sme skúšali s oboma typmi výpočtu pravdepodobnosti, buď mierou $P = P(H|\ell_0)$, alebo mierou $Q = P(H, s|\ell_0)$.

Najprv sme to otestovali na najľahšom vstupe označenom A. Graf s výsledkami testov je na obrázku 5.3.



Obr. 5.3: Závislosť úspešnosti metód od váhy pre vstup A. Hore sú histórie hodnotené mierou P, dole mierou Q.

Prvé, čo si v grafe môžeme všimnúť, je, že metóda hodnotiaca podobnosť sekvencií je v prvom grafe druhá najhoršia a v druhom grafe najlepšia. Dôvodom je to, že táto metóda má tendenciu sekať udalosti na kratšie úseky. Ukážeme si to na skutočnom príklade zo vstupu A. V skutočnej histórii nastala duplikácia atómov `7 8 9`. Súdiac podľa počtu rozdielov vo výsledných sekvenciách a vzorca 4.5 by očakávané časy duplikácie boli postupne 0.0143, 0.0109, 0.0164 časových jednotiek. Priemer týchto hodnôt

je približne 0.01387 a disperzia 0.0022. Z tohto dôvodu udalosť, ktorá duplikuje len atóm `8` s lepším priemerom a nulovou disperziou, dostane oveľa vyššie skóre. Dokonca história, ktorá rozdelí túto udalosť na tri menšie, bude mať pri dobrom určení časov udalostí lepšiu mieru Q ako vzorová. Naopak, miera P závisí najmä od počtu udalostí v histórii, preto v prvom grafe metóda dopadla tak zle.

Ďalej si môžeme všimnúť, že metóda nazvaná minulosť, ktorá heuristicky hodnotí výzor sekvencie pred udalosťou, má veľmi zlé výsledky. Histórie ktoré nájde sú horšie, ako keby sme nepoužili žiadnu metódu. Metóda bola navrhnutá, aby pomáhala rozlišovať, či duplikácia išla zľava doprava alebo naopak. Bohužiaľ nedokáže správne určiť, ktorý úsek bol duplikovaný.

Pri použití miery P sa podaril malý úspech – metóda založená na znižovaní rôznorodosti dosiahla optimálnu pravdepodobnosť. Najlepšia nájdená história bola takmer identická s pôvodnou, líšila sa len v časoch a poradí niektorých udalostí. Poradie nezávislých udalostí má totiž minimálny vplyv na túto mieru. Metóda preferujúca dlhé duplikácie dosiahla tiež pomerne dobré výsledky. Pri použití miery Q sa darilo týmto metódam relatívne dobre, no boli tu isté ťažkosti. Hlavný problém je, že rovnomerné rozmiestnenie udalostí v čase malo nielen negatívny vplyv na výslednú pravdepodobnosť, ale občas aj na samotné postupnosti udalostí. Stávalo sa, že víťazné histórie mali rozdelené udalosti na časti alebo v nich boli nadbytočné delécie len preto, aby zvyšné udalosti mali lepšie časové umiestnenie.

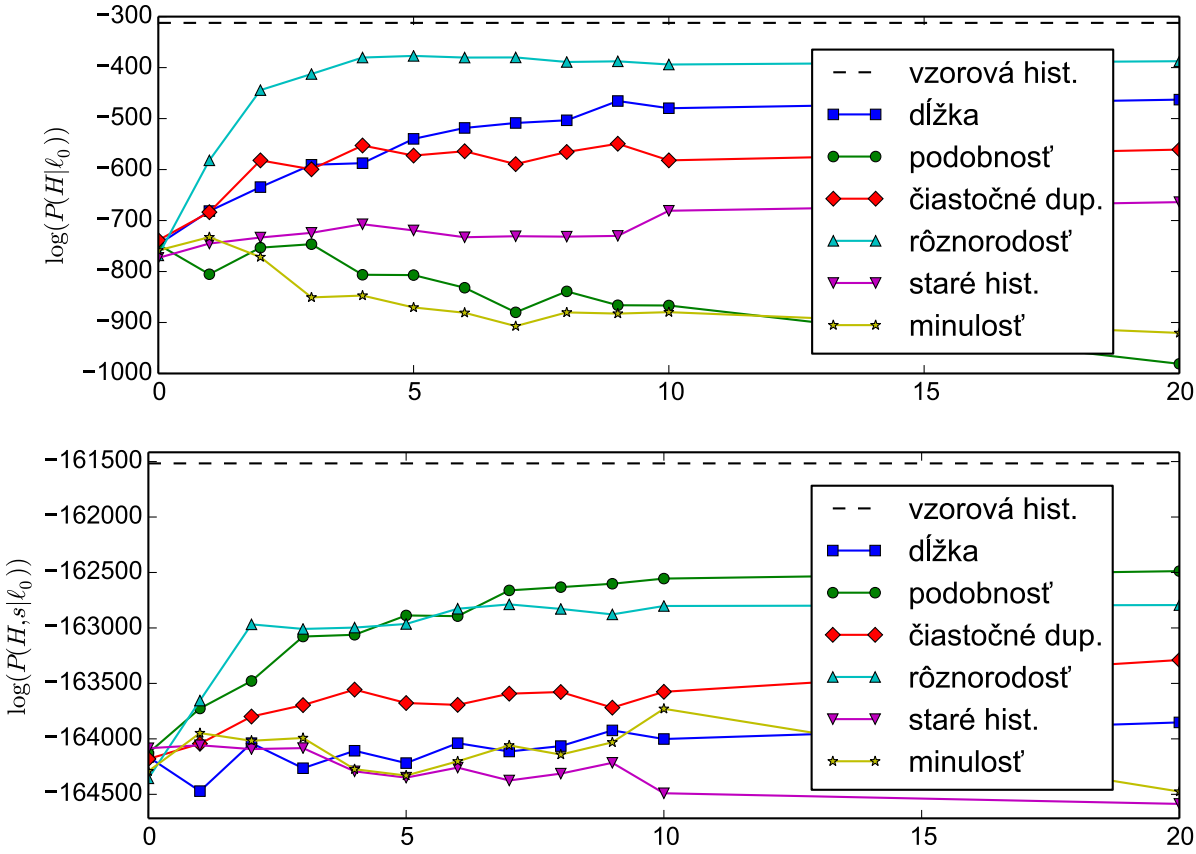
Tiež sme si všimli, že nemá zmysel nastavovať váhy na viac ako 6, pretože celkový výsledok to už nezlepšuje a vysoké váhy znižujú rozmanitosť rekonštruovaných histórií.

Následne sme to isté testovali aj na iných vstupoch. Na obrázku 5.4 vidíme výsledky pre dlhšiu históriu a ťažší vstup C. Výsledky prvých piatich metód sú podobné ako pri predošlom vstupe, hoci rozdiel vidíme napríklad vo vyššej úspešnosti metódy, ktorá penalizuje čiastočné duplikácie.

Šiesta metóda nás prekvapila a zrazu nemala také zlé výsledky ako prvom vstupe. Podľa miery P síce dopadala táto metóda zle aj na ostatných vstupoch, no podľa miery Q na vstupe D dosahovala porovnateľné hodnoty s ostatnými metódami a na vstupe B dokonca lepšie ako väčšina metód a porovnateľne dobre s metódou rôznorodosti. Bohužiaľ to však nebolo spôsobené tým, že by sa nájdená história podobala na vzorovú, skôr naopak. Na začiatku nájdenej histórie bolo veľmi veľa krátkych udalostí, čo spôsobilo lepšie rozmiestnenie časov a vyššiu vierohodnosť stromov lokusov. Tieto výsledky nám skôr naznačujú, že táto metóda nie je dobrá a tiež miera Q nie je vhodná, keď nevieme dobre určiť časy udalostí.

Celkovo sme z týchto meraní usúdili nasledovné:

- Keď chceme použiť metódu skúmajúcu priemer a disperziu očakávaných časov,



Obr. 5.4: Závislosť úspešnosti metód od váhy pre vstup C.

potrebujeme zároveň použiť aj metódu, ktorá pozitívne hodnotí dĺžku duplikovanej sekvencie, alebo tú, ktorá penalizuje čiastočné duplikácie. Zabránime tým zbytočnému sekaniu udalostí na menšie časti. Podobne ani metódu nazvanú minulosť (skúmajúcu celkový vzhľad sekvencie pred udalosťou), nemá zmysel používať samostatne.

- Metóda nazvaná rôznorodosť má z našich metód najväčší pozitívny vplyv na algoritmus, pre jednoduchšie vstupy pomocou nej dokonca dokážeme nájsť najpravdepodobnejšiu históriu.

- Miera P dokáže byť dobrou pomôckou pri hľadaní najlepšej histórie, ale sama o sebe nestačí. Víťazné histórie podľa tejto miery sú zložené zo správnych udalostí, ale často v nesprávnom poradí.

- Miera Q má naopak mnohé nevýhody, pretože časť týkajúca sa génových stromov má príliš veľký vplyv na výsledok. Veľmi často tento spôsob hodnotenia uprednostňuje histórie, ktoré sa menej podobajú na vzorovú. Lokálne sa tiež môže oplatiť rozdeľovať udalosti na menšie úseky. Najmä nie je dobré kombinovať rozmiestňovanie udalostí v pravidelných časových rozostupoch s týmto spôsobom

hodnotenia.

- Najlepší spôsob by bol asi hodnotiť $P(s|H,\ell_0)^x \cdot P(H|\ell_0)$ pre nejaké x. Odhadujeme, že správne x bude okolo 0.2, ale nijako sme takýto spôsob hodnotenia netestovali. Myslíme si totiž, že vo veľkej väčšine prípadov bude vzorová história mať zároveň najmenší počet udalostí spomedzi všetkých histórií. $P(s|H,\ell_0)^x$ by malo slúžiť už len na rozlíšenie histórií s rovnakým počtom udalostí.

Ďalšie testy, ktoré sme robili, kombinovali metódu podobnosti sekvencií, ktorá penalizuje priemer a disperziu očakávaných časov s metódu skúmajúcou dĺžku a metódou, ktorá penalizuje čiastočné duplikácie. Ani takto nebola metóda podobnosti úspešná. Pri použití miery P boli výsledky skoro rovnaké, ako keď sme metódu nepoužili. Pri použití druhej miery bolo jediné zlepšenie v lepšom určení časov udalostí.

Namiesto metódy podobnosti sekvencií sme s tými istými dvoma metódami kombinovali aj metódu nazvanú minulosť. Výsledky záviseli najmä od vstupu. Na vstupe A mala táto kombinácia dosť zlé výsledky, na vstupe C niekedy horšie a niekedy porovnateľné s prípadom kedy sme metódu nepoužili. Na vstupoch B a D bola kombinácia týchto metód lepšia ako každá z týchto metód samostatne, ale nie výrazne. V žiadnom prípade sa však nepodarilo nájsť histórie, ktoré by sa dostatočne podobali na vzorovú.

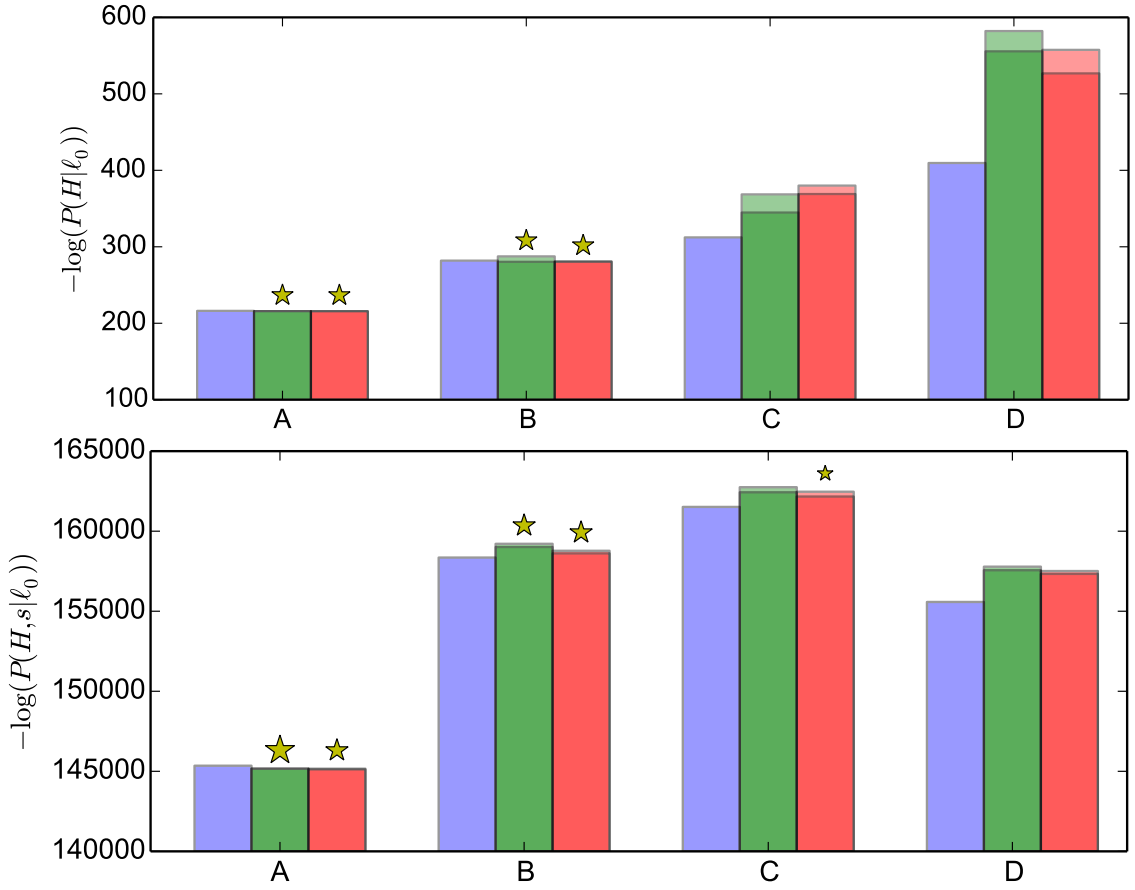
Napokon sme skúšali rôzne kombinácie všetkých metód s cieľom nájsť čo najlepšie celkové nastavenie váh. Najväčšiu váhu dostala metóda rôznorodosti, ktorá najviac pomáhala pri hľadaní správnej histórie. Pri použití miery P mali pozitívny vplyv na výsledok metódy nazvané dĺžka a čiastočné duplikácie.

Pri použití miery Q bolo potrebné zapnúť aj metódu podobnosti, aby algoritmus vedel presnejšie určiť časy udalostí. Veľmi nás prekvapilo, že v kombinácii s ostatnými metódami a dobrým nastavením váh dokázala metóda dobrej minulosti pomôcť. Rozdiel nebol veľmi výrazný, ale bol na všetkých štyroch vstupoch.

Metóda, ktorá uprednostňuje udalosti z nedávno nájdených histórií sa nakoniec vo víťaznej konfigurácii neobjavila. Keď sme ju skúšali pridať, výsledky sa mierne zhoršili, čo si vysvetľujeme tým, že sa znížila rozmanitosť skúšaných histórií, a teda bola menšia pravdepodobnosť nájdenia výnimočne dobrej histórie.

Na obrázku 5.5 môžeme vidieť výsledky pre najúspešnejšie nastavenia. Hodnoty váh pre tieto nastavenia sú v tabuľke 5.2.

Za úspech považujeme to, že pri použití miery P sa nám podarilo nájsť správnu množinu udalostí na prvých dvoch vstupoch. Pri použití miery Q sa nám na najľahšom vstupe podarilo nájsť celú históriu správne, vrátane poradia udalostí. Na druhom najľahšom vstupe sa nájdená história tiež dosť podobala na vzorovú.



Obr. 5.5: Víťazné nastavenia váh otestované na všetkých vstupoch. Prvý (modrý) stĺpec predstavuje vzorovú históriu. Druhý (zelený) a tretí (červený) stĺpec predstavujú histórie nájdené algoritmom, pre dve víťazné kombinácie váh. Vždy celá výška stĺpca je priemer z 5 najlepších histórií v blokoch po 200 iterácií a nižšia tmavšia časť je miera najlepšej histórie zo všetkých 1 000 iterácií. Hviezdička nad stĺpcom znamená, že algoritmom nájdená história bola rovnaká alebo veľmi podobná vzorovej. Subjektívne hodnotená podobnosť je vizualizovaná veľkosťou hviezdičky.

Miera	Stĺpec	v_1	v_2	v_3	v_4	v_5	v_6
P	1. (zelený)	2	0	0	4	0	0
P	2. (červený)	1	0	2	6	0	0
Q	1. (zelený)	2	1	1	5	0	0
Q	2. (červený)	2	1	1	5	0	2

Tabuľka 5.2: Nastavenie váh pre víťazné konfigurácie. Metódy sú očíslované podľa kapitoly 4, čiže postupne: Dĺžka, podobnosť, čiastočné duplikácie, znižovanie rôznorodosti, staré histórie a dobrá minulosť.

Záver

Na začiatku práce sme vysvetlili základné biologické pojmy a popísali použitý pravdepodobnostný model. Ďalej sme popísali problém rekonštrukcie duplikačnej histórie. Spomenuli sme dve predošlé práce v tejto oblasti a algoritmy, ktoré autori používali na riešenie podobného problému. Ukázali sme, ako môžeme využiť pravdepodobnostný model a časti oboch algoritmov v našom probléme.

Podarilo sa nám naprogramovať algoritmus na rekonštrukciu duplikačných histórií. Medzi jeho výhody patrí rýchlosť a vysoká konfigurovateľnosť. Algoritmus je navrhnutý nielen na to, aby sa dali jednoducho meniť váhy skórovacím metódam, ale aby sa dali jednoducho pridať ďalšie skórovacie metódy a vypínať existujúce metódy (napríklad pre dosiahnutie vyššej rýchlosti). Algoritmus je podľa nás vhodný pre skúmanie vplyvu rôznych parametrov na úspešnosť rekonštrukcie histórie.

V našej práci sme skúmali šesť rôznych metód skórovania udalostí a ich kombinácie. Ukázali sme, že metóda z podkapitoly 4.4, ktorá lepšie hodnotí udalosti znižujúce rôznorodosť, je kľúčová pri správnej rekonštrukcii histórie. Ďalej metóda z podkapitoly 4.5, ktorá uprednostňovala udalosti z predošlých histórií, veľmi nepomáha. Metóda hodnotiaca výzor sekvencie pred udalosťou (podkapitola 4.6) mala samostatne veľmi zlé výsledky, najmä na ľahkých vstupoch. Pri vhodnej kombinácii s ostatnými metódami však dokázala pomôcť. Skórovacia metóda, ktorá hodnotila udalosti na základe odhadnutých časov pre jednotlivé atómy (podkapitola 4.2), nemala sama o sebe dobrý vplyv na správnu rekonštrukciu. Jej vedľajším efektom však bolo oveľa presnejšie určenie časov udalostí. To sa ukázalo byť kľúčové, ak použijeme mieru Q spomínanú v kapitole 5. Bohužiaľ mala táto metóda vyššiu časovú zložitosť ako ostatné, čo sa prejavilo v dĺžke behu programu. Zvyšné dve metódy mali mierne pozitívny vplyv na správnosť rekonštrukcie.

Podarilo sa nám nájsť kombinácie váh skórovacích metód so spôsobmi hodnotenia histórie, pre ktoré mal algoritmus pomerne dobré výsledky. Pri miere P boli navyše všetky skórovacie metódy veľmi rýchle. Na ťažších vstupoch sa algoritmu veľmi nedarilo, z čoho vyvodzujeme, že je potrebné nájsť nejakú ďalšiu skórovaciu metódu, ktorá dokáže lepšie rozlíšiť správne udalosti. Prípadne by sa dali spraviť testy s oveľa väčším

počtom iterácií.

Hlavný prínos práce je spomínaný algoritmus na rekonštrukciu duplikačných histórií, rozsiahle merania a podrobný rozbor skórovacích metód.

V budúcnosti by podľa nás bolo dobré preskúmať ďalšie možnosti skórovania a skúsiť nájsť takú metódu, ktorá by pomohla pri rekonštrukcii ťažších vstupov. Myslíme si tiež, že nová metóda by mala využiť informácie zo zarovnaní, a to lepšie, ako metóda pravdepodobnosti. Ďalej je tu možnosť rozšíriť algoritmus, aby dokázal rekonštruovať duplikačnú históriu viacerých organizmov. Informácie o DNA sekvenciách iných organizmov by tiež mohli byť využité v nejakej novej skórovacej metóde.

Okrem toho navrhujeme použiť inú mieru na hodnotenie histórií, pretože obe naše miery mali svoje nedostatky. Kandidát na novú mieru je $P(s|H, \ell_0)^x \cdot P(H|\ell_0)$ pre vhodné x.

Ukázali sme, že pre správnu rekonštrukciu histórie potrebujeme vedieť správne určiť časy udalostí, takže navrhujeme dorobiť algoritmus, ktorý dokáže tieto časy určiť čo najpresnejšie. Nami použité spôsoby určovania časov boli skôr provizórne a mali svoje nedostatky.

Napokon je tu možnosť spraviť komplexnejší algoritmus, ktorý počas svojho behu mení a prispôsobuje váhy jednotlivých skórovacích metód. Napríklad by mohol najprv spraviť veľké množstvo iterácií so zapnutými len rýchlymi skórovacími metódami a mierou P. Takto by navrhol skupinu rozumne vyzerajúcich histórií, ktoré by naďalej zlepšoval s pridanými pomalšími ale presnejšími metódami skórovania udalostí a vhodnejšou mierou.

Literatúra

[And14] Michal Anderle. Časovanie udalostí pri inferencii duplikačných histórii. Bachelor thesis, Comenius University in Bratislava, 2014. Supervised by Tomáš Vinař.

[BBV11] Brona Brejova, Michal Burger, and Tomas Vinar. Automated Segmentation of DNA Sequences with Complex Evolutionary Histories. In Teresa M. Przytycka and Marie-France Sagot, editors, *Algorithms in Bioinformatics, 11th International Workshop (WABI)*, volume 6833 of *Lecture Notes in Computer Science*, pages 1–13, Saarbrücken, Germany, September 2011. Springer.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchinson. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[Kra11] Martin Kravec. MCMC algoritmus na rekonštrukciu duplikačných histórií. Master's thesis, Comenius University in Bratislava, 2011. Supervised by Tomáš Vinař.

[Ora11] Orangutan Genome Sequencing and Analysis Consortium. Comparative and demographic analysis of orang-utan genomes. *Nature*, 469(7331):529–533, 2011.

[VBSS10] Tomas Vinar, Brona Brejova, Giltae Song, and Adam C. Siepel. Reconstructing Histories of Complex Gene Clusters on a Phylogeny. *Journal of Computational Biology*, 17(9):1267–1279, 2010. Early version appeared in RECOMB-CG 2009.

[VVB13] Martina Višňovská, Tomáš Vinař, and Broňa Brejová. DNA Sequence Segmentation Based on Local Similarity. In Tomáš Vinař, editor, *Information Technologies - Applications and Theory (ITAT)*, volume 1003 of *CEUR-WS*, pages 36–43, 2013.