Univerzita Komenského, Bratislava
Fakulta matematiky, fyziky a informatiky

# Časovanie udalostí pri inferencii duplikačných histórií

Bakalárska práca

2014

Michal Anderle

Univerzita Komenského, Bratislava
Fakulta matematiky, fyziky a informatiky

# Časovanie udalostí pri inferencii duplikačných histórií

Bakalárska práca

Študijný program: Informatika
Odbor: 2508 Informatika
Katedra: Katedra informatiky
Vedúci: Mgr. Tomáš Vinař, PhD.

Bratislava, 2014
Michal Anderle


Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Michal Anderle
**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:** Časovanie udalostí pri inferencii duplikačných histórií
*Event Times in Inference of Duplication Histories*

**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 menšie rozsahu (substitúcie, delécie, inzercie), čím vzniká sekvencia so zložitou štruktúrou. Predchádzajúce prístupy v tejto oblasti neuvažovali problém určovania času, kedy nastali jednotlivé udalosti, pri inferencii histórie. Cieľom práce je preštudovať spôsob časovania udalostí pri inferencii fylogenetických stromov a adaptovať tieto prístupy v kontexte inferencie duplikačných histórií.

**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 mojim vedúcim Vinkovi a Broni, že do poslednej chvíle mali viac optimizmu ako ja. Rodine, ktorá ma má, napriek tomu, že ma vidí len zriedka, stále rada a podporuje ma. Trojstenu za to, že som aký som. Všetkým mojim kamarátom, ktorí ma povzbudzovali, podporovali a taktiež rozptylovali. A nakoniec chcem poďakovať Peti a Janovi, že sú vždy ochotní pomôcť a sú tými najlepšími kamarátmi, akých si človek môže želať.

# Abstrakt

Nosičom genetickej informácie v bunkách živých organizmov je DNA. Zaujímavou otázkou je, či vieme podľa DNA súčasných druhov určiť, ako vyzerala DNA sekvencia ich predkov. V rámci evolúcie musíme rátať s náhodnými udalosťami, ktoré ovplyvňovali danú sekvenciu, či už sú to jednoznakové substitúcie alebo väčšia evolučné udalosti ako duplikácie a delécie. Práve tie vytvárajú duplikačné histórie popisujúce postupný vývoj genómu. Dôležitým prvkom týchto histórií je načasovanie jednotlivých udalostí. V našej práci si predstavíme model na výpočet pravdepodobnosti duplikačných histórií a postup, ktorý sa používa na upravovanie časov vo fylogenetickom strome s cieľom získania stromu s väčšou pravdepodobnosťou. Tento prístup upravíme tak, aby bol aplikovateľný na časovanie udalostí v duplikačných históriách. Našu implementáciu potom otestujeme na simulovaných dátach a dosiahnee pozitívne výsledky ukazujúce, že náš prístup zlepšuje duplikačné histórie a má potenciál na využitie pri reálnych dátach.

**Kľúčové slová:** duplikačná história, časovanie udalostí, fylogenetický strom

# Abstract

The DNA is the carrier of all genetic information that we can find in cells of living organisms. We can ask ourselves whether it is possible to determine ancestral DNA sequences from sequences found in extant species. When we consider the evolutionary process, we need to take into account random events which change the DNA sequence. On one side, we have small substitutions of one character by another; on the other side, we have large evolutionary events such as duplications and deletions. These evolutionary events create duplication histories representing an evolution of the genome. An important part of these histories is the exact time of individual events. In this thesis, we describe an existing model estimating the likelihood of duplication histories. We also outline a procedure for improving the likelihood of phylogenetic trees by adjusting their edge times. As our main contribution, we introduce a modification of this procedure that operates on duplication histories. Finally, we evaluate our implementation of this procedure on simulated data sets. The results indicate that our method can be useful in improving duplication histories in real data.

**Keywords:** duplication history, event times, phylogenetic tree

# Obsah

# Zoznam obrázkov

# Zoznam tabuliek

# Úvod

S príchodom nových technológií sa začali rozvíjať všetky vedné odbory, vrátane biológie. Zrazu bolo možné sekvenovať DNA z buniek živých organizmov. Tento pokrok však priniesol nové otázky. Napríklad nás môže zaujímať, či vieme zistiť z DNA súčastných živočíchov, DNA ich predkov. DNA totiž podliehala evolúcii a aj keď je evolúcia náhodný proces, riadi sa istými pravidlami. Spôsobov ako sa môže meniť DNA však nie je tak veľa. Najväčšou takouto zmenou je nakopírovanie alebo odstránenie časti genómu. Tieto udalosti potom vytvárajú sekvencie so zložitou vnútornou štruktúrou, ktorej odhalenie by mohlo viesť k zisteniu ancestrálnej DNA sekvencie [6, 9].

Snahou bioinformatikov je pre namerané sekvencie zisťovať ich duplikačnú históriu. Teda postupnosť udalostí ako sú delécie alebo duplikácie, ktorých výsledkom bola naša sekvencia. Dôležitou súčasťou týchto histórií však nie je len samotný tvar udalostí, ale taktiež čas, kedy sa v histórií vyskytli. Avšak, napriek tomu, že tieto časy výrazne ovplyvňujú pravdepodobnosť, že naša história je správna, algoritmy, ktoré vytvárajú duplikačné histórie [5], majú veľký problém s nachádzaním správnych časov pre jednotlivé udalosti. V našej práci sa teda zameriame práve na tento podproblém. Budeme sa snažiť vylepšiť danú duplikačnú históriu tým, že budeme meniť časy, kedy nastali jednotlivé udalosti.

Podobný problém nastáva aj pri jednoduchšom modeli fylogenetických stromov. Existujú algoritmy, ktoré sa snažia zlepšovať daný fylogenetický strom len zmenou dĺžok hrán v ňom [4, 8, 11]. V rámci nich sa používajú rôzne numerické metódy, akou je aj Newton-Raphsonova metóda[1, 7] na hľadanie koreňa funkcie. Náš problém je však komplikovanejší, lebo sa snažíme optimalizovať niekoľko, od seba závislých, stromov naraz. V našej práci si ukážeme ako sa dá prístup z fylogenetických stromov upraviť a aplikovať na problém časovania udalostí v duplikačnej histórii.

Naše riešenie sme takisto implementovali v programovacom jazyku C++ a túto implementáciu odskúšali na simulovaných dátach. Výsledky ukázali, že náš prístup má

veľký potenciál, pretože sa nám podarilo zlepšiť takmer všetky duplikačné histórie na vstupe.

Zvyšok práce je štrukturovaný nasledovne:

V 1. kapitole si presnejšie zadefinujeme problém, ktorý budeme riešiť a zavedieme si niekoľko kľúčových pojmov. V 2. kapitole si ukážeme prístup, ktorý sa používa na úpravu hrán vo fylogenetických stromoch a predstavíme si algoritmy a postupy, ktoré budeme používať aj v našej práci. V 3. kapitole upravíme prístup z fylogenetických stromov tak, aby bol aplikovateľný aj na duplikačné histórie. V poslednej 4. kapitole si povieme niečo o implementácií konkrétneho riešenia a spravíme niekoľko testov na umelých dátach, aby sme videli efektívnosť nášeho riešenia. Implementáciu tohoto riešenia môžete nájsť na `ksp.sk/~zaba/anderle_bakalarska_praca.zip`.

# Kapitola 1

# Rekonštrukcia duplikačných histórií

V tejto kapitole popíšeme problém, ktorý sa snažíme riešiť, motiváciu z ktorej tento problém vychádza a takisto základné pojmy bioinformatiky, ktoré budem používať naprieč celou prácou a je preto nevyhnutné, aby sme si ich sformulovali.

## 1.1 Biologická motivácia problému a proces evolúcie

V bunkách všetkých živých organizmov sa nachádza DNA, ktorá nesie genetickú informáciu o danom organizme. Pomocou zložitých biotechnických postupov a metód sme schopný z buniek vyextrahovať DNA a transformovať ju na postupnosť písmen z abecedy $\{A, C, G, T\}$. Každý znak tejto postupnosti zastupuje jeden nukleotid – základnú stavebnú jednotku DNA. V DNA sa vyskytujú štyri druhy nukleotidov a to adenín, cytozín, gaunín a tymín.

Darwinova evolučná teória predpokladá, že každé dva organizmy vznikli zo spoločného predka. Naskytá sa teda otázka, či je možné, na základe osekvenovanej DNA, zistiť, ako vyzerala DNA tohoto spoločného predka, alebo dokonca zistiť, ako sa táto DNA časom vyvíjala.

Problémom však je, že DNA sa z generácie na generáciu mení. A tieto zmeny sú častokrát náhodné a nekontrolovateľné. Ak by sme predpokladali len nezávislé, jednoznakové zmeny jedného nukleotidu na iný, problém by nebol tak ťažký. Existuje množstvo pravdepodobnostných modelov (jeden z nich, ktorý budeme používať, popíšeme v ďalšej kapitole), ktoré vedia pre daný znak povedať najpravdepodobnejší ancestrálny znak, ktorý sa tam mohol v minulosti nachádzať.

V našom modeli evolúcie sa však vyskytujú aj takzvané evolučné alebo duplikačné udalosti. Sú to zložitejšie zmeny DNA, ktoré na rozdiel od jednoznakových substitúcií, ovplyvňujú viacero nukleotidov naraz. V našej práci budeme uvažovať nasledovné evolučné udalosti:

- Duplikácia – skopírovanie časti DNA sekvencie a jej vsunutie na inú pozíciu.
- Reverzná duplikácia – podobne ako duplikácia, ale duplikovaná časť sa na nové miesto kopíruje odzadu a komplementárne. To znamená, že $A$ sa zmení na $T$, $C$ sa zmení na $G$, $G$ sa zmení na $C$ a $T$ sa zmení na $A$.
- Delécia – vymazanie súvislej časti DNA sekvencie.

Poslednou udalosťou, ktorú však v našej práci nebudeme uvažovať, je speciácia. Táto udalosť vyjadruje rozdelenie oraganizmu na dve nezávislé skupiny, ktoré sa následne vyvíjali samostatne bez toho, aby sa navzájom ovplyvňovali. Dá sa na to taktiež pozerať tak, že ancestrálna sekvencia sa rozdelí na dve totožné sekvencie, ktoré sa ďalej menia nezávisle. Treba si uvedomiť, že ak nastala speciácia, ale jeden z druhov neprežil do súčastnosti, nie je možné túto speciáciu detekovať, lebo nemáme sekvenciu DNA, ktorá prislúcha mŕtvemu druhu.

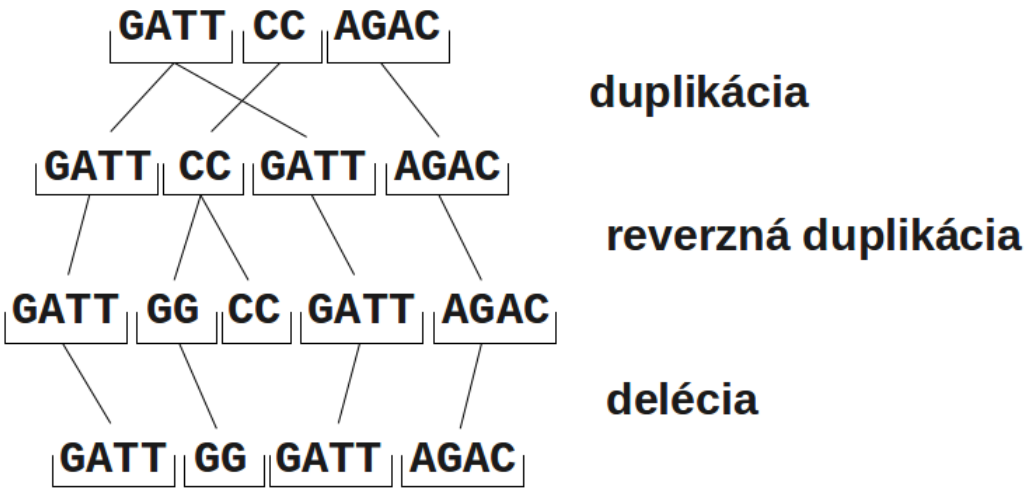
Zo získaných sekvencií DNA sa teda snažíme zistiť, aké udalosti nastali v priebehu generácií a následne tieto udalosti obracať, napríklad odstránením naduplikovanej časti, a získavať tak ancestrálnu DNA.

## 1.2 Duplikačná história, atóm a stromy atómov

Aby sme mohli presne zadefinovať problém, ktorý sa snažíme riešiť, potrebujeme si zaviesť nasledovné tri pojmy.

### 1.2.1 Duplikačná história

V predchádzajúcej podkapitole sme si ukázali duplikačné udalosti. Takisto sme si povedali, že sa snažíme zistiť aké udalosti nastali, aby sme z pôvodnej ancestrálnej DNA dostali súčasnú DNA. Duplikačná história je návodom, ktorý hovorí o zmene ancestrálnej DNA. Obsahuje zoznam udalostí, ich typov a času, kedy tieto udalosti nastali, usporiadaných do stromu (cesty), kde dĺžky hrán reprezentujú čas, ktorý prešiel medzi dvoma udalosťami.




Obr. 1.1: Jednoduchá duplikačná história s vynechanými časmi udalostí

Na obrázku 1.1 môžeme vidieť príklad duplikačnej histórie pozostávajúcej z troch udalostí. Začiatočná ancestrálna sekvencia mala tvar `GATTCCAGAC`. Ako prvá sa v histórii vyskytla duplikácia časti `GATT`, čím sme dostali sekvenciu `GATTCCGATTAGAC`. Následne sa udiala reverzná duplikácia, ktorá nakopírovala časť `CC` ako `GG`, čo je príslušný reverzný reťazec. Nakoniec, posledná udalosť, delécia, odstránila časť `CC`.

### 1.2.2 Atóm

V duplikačnej histórii rozoberáme iba veľké duplikačné udalosti a abstrahujeme od jednoznakových substitúcií. Predpokladáme teda, že operácie, ktoré sa v nej vyskytujú, upravovali veľké kusy DNA naraz. Nebolo by teda dobré, aby sme ako najmenšiu jednotku brali jeden nukleotid, pretože nevieme zistiť, či viacnásobný výskyt tohoto nukleotidu vznikol duplikáciou, alebo majú tieto nukleotidy rôzneho predka.

Keďže sa však vždy upravovali väčšie časti DNA, dá sa predpokladať, že susedia nukleotidov so spoločným ancestrálnym nukleotidom vyzerajú podobne, lebo sa nakopírovali spolu s ním. Ak by sme teda našli dve časti DNA, ktoré sú veľmi podobné, dá sa predpokladať, že vznikli duplikáciou.

Chceli by sme teda nasekať DNA na také segmenty, aby sa tieto segmenty dali rozdeliť do skupín, kde ľubovoľné dva segmenty z rovnakej skupiny sú si veľmi podobné a ľubovoľné dva segmenty z rôznych skupín sú si podobné veľmi minimálne. Takéto segmenty nazveme atomické segmenty, alebo zjednodušene atómy.

Atómy sú pre nás v rámci duplikačnej histórie najmenšie, nedeliteľné úseky DNA sekvencie, ktoré sú duplikačnou udalosťou ovplyvnené vždy celé. Udalosti teda majú začiatky a konce na hraniciach dvoch atómov.

Dôvod, prečo sme celý čas vraveli, že atómy patriace do jednej triedy si majú byť veľmi podobné je ten, že na jednotlivé kópie pôvodného ancestrálneho atómu pôsobili jednoznakové mutácie. Je teda pravdepodobné, že ak niekedy boli rovnaké, teraz sa od seba líšia len málo. A tak isto, ak dva atómy boli rozdielne, je malá šanca, že náhodnými substitúciami príliš spodobnejú.

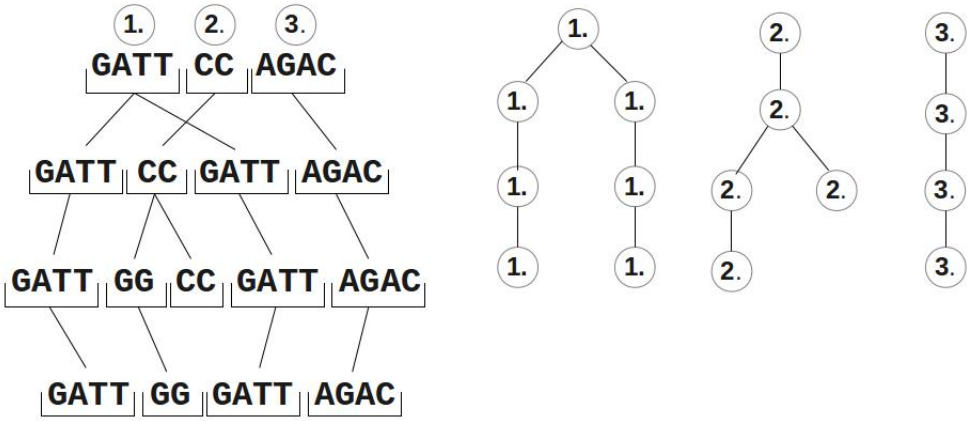
Našťastie, nasekanie DNA na jednotlivé atómy nemusíme riešiť my, pretože existujú algoritmy, ktoré vedia tieto atómy z DNA získať [2, 10]. Takisto si všimnime, že atómy nevyplývajú zo žiadnych biologických procesov, sú len technickou pomôckou na ľahšiu prácu so vstupom.

### 1.2.3 Stromy atómov

Ukázali sme si, že vieme našu DNA nasekať na atómy, ktoré sa dajú rozdeliť do tried podobnosti. O každej triede naviac predpokladáme, že vznikla z jedného ancestrálneho atómu. Otázkou však je, ako vyzerala duplikácia tohoto ancestrálneho atómu, že sme dostali práve našu triedu atómov.

Tak, ako pri duplikačných históriach, si môžeme reprezentovať toto delenie ako strom, ktorého koreň je ancestrálny atóm a dĺžky hrán určujú časy, ktoré prešli od jedného rozdelenia po ďalšie. Takýto strom budeme nazývať strom atómov, alebo atómový strom. Na rozdiel od histórie, tentokrát už nedostaneme len cestu, ale náš strom bude mať taký počet listov, koľko atómov patrí do tejto triedy podobnosti, lebo každý list predstavuje jeden atóm z tejto triedy.

Všimnime si, že tento strom má priamy súvis s duplikačnou históriou prislúchajúcou k tejto sekvencii, pretože ak máme niekde v atómovom strome vrchol, ktorý má dvoch synov, znamená to, že tento ancestrálny atóm sa musel v danom momente duplikovať, a preto sa v histórii musí nachádzať duplikácia. Naopak, ak sa v histórii objaví udalosť ovplyvňujúca tento typ atómu, musí sa v strome atómov na príslušnom mieste objaviť zodpovedajúci vrchol. Túto súvislosť si môžeme všimnúť aj na obrázku 1.2.




Obr. 1.2: Duplikačná história obsahujúca tri triedy atómov a im prislúchajúce atómové stromy

## 1.3 Časovanie udalostí duplikačnej histórie

Z biologické hľadiska by nás zaujímalo, ako vyzerala ancestrálna DNA pre nami získanú sekvenciu. Okrem toho by sme chceli vybudovať aj duplikačnú históriu, ktorá by zodpovedala postupnému vývoju tejto DNA. Problém však je, že všetky evolučné procesy sú náhodné a nedajú sa deterministicky určovať.

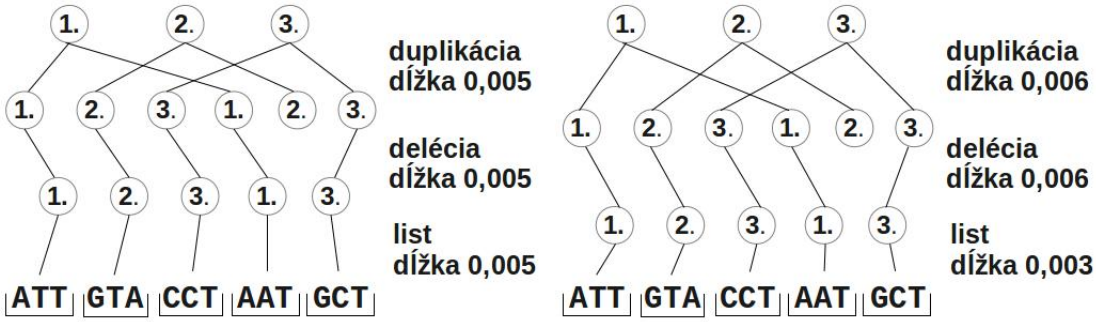
Namiesto toho, aby sme hľadali tú správnu duplikačnú históriu, teda tú, ktorá zodpovedá skutočnému priebehu, budeme hľadať takú duplikačnú históriu, ktorá má vzhľadom na nami zvolený model najväčšiu pravdepodobnosť výskytu. Síce takáto história sa od tej skutočnej môže líšiť, je ale pravdepodobné, že sa tej skutočnej podobá a môže nám tak dať cenné informácie.

Takýto problém je samozrejme veľmi komplexný, a preto ho nebudeme riešiť celý. Zameriame sa len na jeho časť, konkrétne na časovanie udalostí. To znamená, že na vstupe budeme očakávať hotovú duplikačnú históriu, v ktorej budeme meniť časy, kedy nastali jednotlivé udalosti, teda meniť dĺžky hrán tak, aby nová história bola čo najpravdepodobnejšia a nezmenili sme celkový čas, ktorý pokrýva. Chceme teda zachovať súčet dĺžok hrán v histórii.

Popíšme si teraz vstup a výstup podrobnejšie. Na vstupe budeme očakávať sekvenciu časti DNA, pre ktorú hľadám ancestrálnu sekvenciu. Táto sekvencia bude navyše rozsegmentovaná na jednotlivé atómy a tieto atómy budú rozdelené na triedy podobnosti. Pre každú triedu budeme mať dané ešte vzájomné zarovnania atómov v nej.

Navyše Ján Hozza [5] rieši vo svojej bakalárskej práci prvú časť nášeho problému. Jeho algoritmus generuje pre sekvenciu DNA, atómy a zarovnania, duplikačnú históriu, ktorá zodpovedá danému vstupu. Táto história však rozdeľuje časy udalostiam rovnomerne. Na vstupe budeme teda očakávať aj výstup jeho programu, teda validnú duplikačnú históriu [6, 9].

Cieľom našej práce bude upraviť históriu na vstupe tak, aby sme dostali históriu s vyššou pravdepodobnosťou. Pri tomto upravovaní máme zakázané meniť poradie udalostí a taktiež celkový čas, ktorý daná duplikačná história pokrýva. Dovolené je teda posúvať čas niektorých udalostí v rámci rozmedzia medzi predchádzajúcou a nasledujúcou udalosťou. Na výstup máme potom zapísať túto novú, upravenú históriu.




Obr. 1.3: Príklad možnej vstupnej a výstupnej histórie

Na obrázku 1.3 môžeme vidieť, ako vyzerá vstupná a výstupná história. Všimnime si, že nepoznáme konkrétne hodnoty ancestrálnych atómov, len ich zaradenie do jednotlivých tried, v tomto prípade troch možných. Vidíme, že výstupná história sa líši len dĺžkou jednotlivých hrán.

# Kapitola 2

# Inferencia dĺžok hrán vo fylogenetickom strome

Cieľom našej práce je upravovať dĺžky hrán histórie tak, aby sme zvýšili jej pravdepodobnosť v danom modeli. Prístup, ktorý sme si zvolili je odvodený od prístupu, ktorý sa používa na upravovanie dĺžok hrán vo fylogenetickom strome.

V tejto kapitole teda najskôr predstavíme pravdepodobnostný model, s ktorým pracujeme, a taktiež základné algoritmy, ktoré sa používajú pri inferencii dĺžok hrán vo fylogenetickom strome. Na záver uvedieme postup, akým sú upravované dĺžky hrán fylogenetického stromu a od ktorého sa bude odvíjať naša práca.

## 2.1 Jukes-Cantorov model substitúcií

Základnou biologickou zmenou genómu je substitúcia, teda zmena jedného nukleotidu na iný. Keďže tento proces výrazne ovplyvňuje vierohodnosť fylogenetického stromu, je potrebný model evolúcie, ktorý bude tento proces popisovať. V našej práci budeme používať Jukes-Cantorov model substitúcií [3], ktorý je síce pomerne jednoduchý, ale na naše potreby postačujúci.

Jukes-Cantorov model predpokladá, že jediná operácia, ktorá môže nastať, je zmena jedného nukleotidu na iný. Takisto predpokladá, že tieto zmeny sa dejú na rôznych nukleotidoch nezávisle a navzájom sa neovplyvňujú. Navyše je tento model multiplikatívny, teda predpokladá, že pravdepodobnosť mutácie závisí len od aktuálneho znaku a nie od predchádzajúcich.

Označme si $P(b|a,t)$ pravdepodobnosť, že sa nukleotid $a$ zmení za čas $t$ na nukleotid $b$. Z nezávislosti zmien teda vyplýva, že ak máme dve rovnako dlhé sekvencie $x$ a $y$, tak

$P(x|y,t) = \prod_u P(x_u|y_u,t)$, kde $u$ postupne indexuje jednotlivé nukleotidy sekvencie. Z multiplikatívnosti modelu tiež môžeme odvodiť, že ak máme dva časy $t_1$ a $t_2$, tak platí rovnica $P(b|a,t_1+t_2) = \sum_{x\in\Sigma} P(x|a,t_1)\cdot P(b|x,t_2)$.

Ak chceme pracovať s pravdepodobnosťami $P(a|b,t)$ pre všetky $a$ a všetky $b$ naraz, môžeme ich zhrnúť do matice zmien veľkosti $4\times 4$, ktorá bude mať nasledovný tvar:

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

Je zjavné, že vlastnosť multiplikatívnosti sa prenesie aj do takéhoto zápisu a $S(t_1)S(t_2) = S(t_1+t_2)$.

Jukes-Cantorov model ďalej uvažuje maticu rýchlostí zmien $R$, kde každý typ nukleotidu podlieha zmene s rovnakou rýchlosťou $\alpha$. Matica $R$ vyzerá takto:

$$\begin{pmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{pmatrix}$$

Poďme si teraz určiť, ako vyzerajú prvky matice $S(t)$. Pre veľmi malý čas $\epsilon$ si môžeme maticu $S(t)$ aproximovať ako $I+R\epsilon$, kde $I$ je matica identity. Z multiplikatívnosti vyplýva, že $S(t+\epsilon) = S(t)S(\epsilon) \simeq S(t)(I+R\epsilon)$. Túto rovnicu môžeme ďalej upraviť na tvar $(S(t+\epsilon)-S(t))/\epsilon \simeq S(t)R$. Ak teraz pošleme $\epsilon$ limitne k nule, dostávame diferenciálnu rovnicu $S'(t) = S(t)R$.

Následne už nie je problém túto rovnicu vyriešiť. Zo symetrickosti matice $R$ predpokladáme, že aj matica $S(t)$ bude symetrická. Označme si teda $r_t$ diagonálny a $s_t$ nediagonálny prvok matice $S(t)$. Zistíme, že $r_t = \frac{1}{4}(1+3e^{-4\alpha t})$ a $s_t = \frac{1}{4}(1-e^{-4\alpha t})$.

Tieto dve hodnoty, spolu s maticou $R$, tvoria Jukes-Cantorov model mutácií. Je dobré si všimnúť, že náš model zodpovedá našim intuitívnym predstavám, lebo pre $t=0$, $S(t)=I$ a pre $t$ idúce do nekonečna sa $r_t = s_t \to \frac{1}{4}$.

## 2.2 Felsensteinov algoritmus

Jednoznakové substitúcie výrazne ovplyvňujú výslednú pravdepodobnosť či už histórie alebo fylogenetického stromu. V predchádzajúcej podkapitole sme uviedli model, ktorý tieto substitúcie popisuje. Problém, ktorý potrebujeme riešiť je však zložitejší.

Postupom evolúcie sa totiž atómy duplikovali do rôznych častí DNA, kde sa mutovali nezávisle od seba. My máme na vstupe ich súčastný stav, predpokladáme však, že každá trieda atómov vznikla zo spoločného ancestrálneho atómu. Chceli by sme teda vedieť, kedy nastali jednotlivé duplikácie – vytvoriť si strom, ktorý by zobrazoval, ako sa delili pôvodné ancestrálne atómy. Felsensteinov algoritmus [3] slúži na to, aby sme vypočítali pravdepodobnosť, že súčasné sekvencie vznikli podľa vyššieuvedeného substitučného modelu za predpokladu, že s vyvíjali podľa konkrétneho atómového stromu.

### 2.2.1 Dve sekvencie

Najskôr predpokladajme, že máme len dve zarovnané sekvencie $x^1$ a $x^2$. Na nich si predvedieme, princíp fungovania Felsensteinovho algoritmu a v ďalšej sekcii si tento proces rozšírime na ľubovoľný počet sekvencií.

Majme teda sekvencie $x^1$ a $x^2$, každá obsahujúca $n$ nukleotidov a jediný možný strom $T$, ktorý im môže prislúchať – koreň (ancestrálny atóm) a dva listy ($x^1$ a $x^2$). Dĺžku hrany do listu $x^1$ si označíme $t_{x^1}$ a dĺžku hrany do $x^2$ si označíme $t_{x^2}$. $P(x^1, x^2|T, t_{x^1}, t_{x^2})$ bude označovať pravdepodobnosť, že ak sme mali ancestrálny atóm, ktorý sa duplikoval podľa stromu $T$ a časy hrán sú $t_{x^1}$ a $t_{x^2}$, tak dostaneme hodnoty $x^1$ a $x^2$.

Keďže jednotlivé nukleotidy v našich sekvenciách sú od seba nezávislé (tak ako nám hovorí Jukes-Cantorov model), tak pre každý z nich môžeme pravdepodobnosť rátať samostatne. Výsledná pravdepobnosť teda bude súčinom pravdepodobností pre jednotlivé nukleotidy:

$$P(x^1, x^2|T, t_{x^1}, t_{x^2}) = \prod_{u=1}^{n} P(x_u^1, x_u^2|T, t_{x^1}, t_{x^2}) \tag{2.1}$$

Zamerajme sa teraz na nukleotid na pozícii $u$. V listoch nášho stromu teda máme $x_u^1$ a $x_u^2$ a koreňu priradíme nukleotid $a$. Teraz už nie je problém danú pravdepodobnosť vyrátať: z nukleotidu $a$ sa totiž musel za čas $t_{x^1}$ stať nukleotid $x_u^1$, čomu podľa Jukes-Cantorovho modelu zodpovedá hodnota $P(x_u^1|a, t_{x^1})$ a za čas $t_{x^2}$ sa musel zmeniť na $x_u^2$, čomu zodpovedá hodnota $P(x_u^2|a, t_{x^2})$. Navyše ešte musíme uvažovať, nakoľko je

pravdepodobné, že v koreni bol nukleotid $a$, náš zjednodušený model však predpokladá, že táto pravdepodobnosť je rovnomerne rozdelená medzi všetky nukleotidy, a teda rovná $\frac{1}{4}$. Dostávame rovnicu:

$$P(x_u^1, x_u^2, a|T, t_{x^1}, t_{x^2}) = \frac{1}{4} P(x_u^1|a, t_{x^1}) P(x_u^2|a, t_{x^2}) \tag{2.2}$$

My však nevieme, ako vyzeral ancestrálny atóm, a teda ani akú hodnotu máme dosadiť do koreňa. Musíme teda vyskúšať všetky a výsledné pravdepodobnosti sčítať. Dostávame teda výsledný vzorec:

$$P(x_u^1, x_u^2|T, t_{x^1}, t_{x^2}) = \sum_{a \in \Sigma} P(x_u^1, x_u^2, a|T, t_{x^1}, t_{x^2}) \tag{2.3}$$

### 2.2.2 Ľubovoľný počet sekvencií

Rozšírme si predchádzajúce úvahy na ľubovoľný počet sekvencií. Majme $k$ zarovnaných sekvencií $x^1$ až $x^k$, tvorených $n$ nukleotidmy a strom $T$, ktorý má $n$ listov. Označme si $par(i)$ otca vrcholu $i$ v strome $T$ a dĺžku hrany vedúcu z vrchola $i$ do jeho otca si označme $t_i$. Opäť sa zamerajme len na nukleotidy na pozícii $u$.

Priraďme listom stromu $T$ čísla od 1 po $k$ a zvyšné vrcholy očíslujme od $k+1$ po $2k-1$, pričom $2k-1$ je koreňom celého stromu. Opäť sa pokúsime priraďovať jednotlivým vrcholom stromu, ktoré nie sú listy, všetky možné nukleotidy. Tieto priradenia si označme $a^i$ pre $i$-ty vrchol. Pravdepodobnosť pre nejaký vyplnený strom je súčin pravdepodobností na jednotlivých hranách. A výslednú pravdepodobnosť dostaneme súčtom týchto pravdepodobností cez všetky možné priradenia $a^i$.

Výslednú pravdepodobnosť, ktorú sa snažíme vyrátať si vyjadríme ako

$$P(x^1 \dots x^k|T, t_1 \dots t_{2k-2}) = \sum_{a^{k+1}, a^{k+2} \dots a^{2k-1} \in \Sigma} \frac{1}{4} \prod_{i=k+1}^{2k-2} P(a^i|a^{par(i)}, t_i) \prod_{i=1}^{k} P(x_u^i|a^{par(i)}, t_i) \tag{2.4}$$

### 2.2.3 Implementácia

Samozrejme, vyššie uvedenú formulu nemôžeme vyčíslovať priamo. Časová zložitosť takéhoto prístupu by totiž bola exponencionálna. Felsensteinov algoritmus namiesto toho používa prístup dynamického programovania. Postupne pre všetky podstromy ráta pravdepodobnosť $P(L_\ell|a)$, teda pravdepodobnosť, že všetky listy v podstrome s

koreňom vo vrchole $\ell$ vznikli zo znaku $a$. Túto pravdepodobnosť však vieme rekurzívne spočítať s hodnôť $P(L_i|b)$ a $P(L_j|c)$, kde $i$ a $j$ sú synovia vrcholu $\ell$ a $b$, $c$ sú ľubovoľné znaky. Tento vzťah popisuje nasledovný vzorec:

$$P(L_\ell|a) = \sum_{b,c\in\Sigma} P(b|a,t_i)P(L_i|b)P(c|a,t_j)P(L_j|c) \tag{2.5}$$

Aby rekurencia funkcia fungovala správne, musíme určiť nejaké začiatočné hodnoty. Tie nás zaujímajú len v listoch, lebo len tie nemajú ďalších synov, a preto nie je možné použiť našu rekurenciu. Hodnota $P(L_\ell|a)$ je v liste $\ell$ rovná 1, ak sa $x^\ell$ rovná $a$. V opačnom prípade je táto pravdepodobnosť 0.

V zarovnaniach, ktoré dostávame ako vstupné dáta sa môže stať, že niektorá hodnota nukleotidu nie je nameraná, alebo je inak poškodená, a nevieme určiť, aký znak sa nachádzal na danej pozícii. V takom prípade sa chováme, akoby tam boli možné všetky štyri znaky a pravdepodobnosť $P(L_\ell|a)$ nastavíme na 1 pre všetky hodnoty $a$.

Keď máme rekurenciu a aj začiatočné hodnoty, môžeme prechádzať stromom v post-order poradí a rátať čiastočné pravdepodobnosti od listov ku koreňu. Výslednú pravdepodobnosť daného stromu zistíme ako súčet čiastočných pravdepodobností pre koreň:

$$\sum_{a\in\Sigma} \frac{1}{4} P(L_{2k-1}|a) \tag{2.6}$$

Ukážme si pseudokód ukazujúci ako funguje Felsensteinov algoritmus.

`Inicializácia:`
    `Nastav premennú` $\ell = 2k-1$
`Rekurzia na rátanie hodnoty` $P(L_\ell|a)$ `pre všetky` $a$`:`
    `Ak je vrchol` $\ell$ `list:`
        $P(L_\ell|a) = 1$ `ak` $a = x^\ell$, $P(L_\ell|a) = 0$ `ak` $a \neq x^\ell$
    `Ak vrchol` $\ell$ `nie je list:`
        `Rekurzívne vypočítaj hodnoty` $P(L_i|a)$ `a` $P(L_j|a)$ `pre všetky` $a$
        `Vypočítaj` $P(L_\ell|a) = \sum_{b,c\in\Sigma} P(b|a,t_i)P(L_i|b)P(c|a,t_j)P(L_j|c)$
`Ukončenie:`
    `Vráť hodnotu` $\sum_{a\in\Sigma} \frac{1}{4} P(L_{2k-1}|a)$

## 2.3 Newton-Raphsonova metóda

Skôr, ako si predstavíme spôsob upravovania hrán vo fylogenetickom strome, potrebujeme si predstaviť posledný algoritmus, ktorý budeme potrebovať pri jeho vysvetľovaní. Síce na pohľad nemá z našim problémom súvis, ako uvidíme v ďalšej podkapitole, bude veľmi podstatný. Tento algoritmus je Newton-Raphsonova metóda na hľadanie koreňa funkcie $f(x)$ [1, 7].

### 2.3.1 Popis metódy

Nájsť koreň funkcie $f$ znamená hľadať takú hodnotu $x$, že $f(x) = 0$. Toto môže byť pri mnohých funkciách problém a nájsť koreň môže byť veľmi komplikované. Častokrát nám však vystačí aj dostatočne dobrá aproximácia koreňa. Newton-Raphsonova metóda sa snaží použiť práve takýto prístup, aproximovaním počiatočného odhadu na stále lepšie a lepšie odhady za pomoci dotyčníc grafu $f$.

Na aproximovanie koreňa funkcie $f$ začnime s úvodným odhadom $x_0$ takým, že $f(x_0) \neq 0$. Skúsime teraz nájsť lepší odhad $x_1$ tak, že zoberieme dotyčnicu k funkcii $f$ v bode $(x_0, f(x_0))$ a za hodnotu $x_1$ zvolíme priesečnicu tejto dotyčnice s osou $x$. Vieme, že sklon dotyčnice funkcie v nejakom bode je určený jej deriváciou. Máme teda rovnicu:

$$f'(x_0) = \frac{\Delta y}{\Delta x} = \frac{f(x_0) - 0}{x_0 - x_1} \tag{2.7}$$

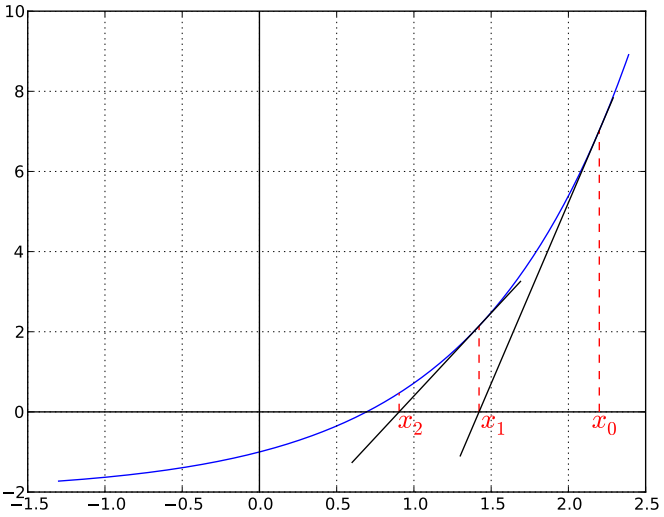
Úpravou tejto rovnice dostaneme vyjadrenie pre hodnotu $x_1$:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{2.8}$$

Samozrejme, tento postup môžeme opakovať ďalej a dostaneme postupnosť aproximácií $x_0, x_1 \ldots x_n$, kde každá nasledujúca sa vypočíta ako $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$. Ak si naviac zvolíme požadovanú presnosť $\epsilon > 0$, môžeme postupne aproximovať stále presnejší koreň, až kým $|x_{i+1} - x_i| < \epsilon$.

### 2.3.2 Konvergencia Newton-Raphsonovej metódy

Táto metóda sa bohužiaľ v mnohom spolieha na to, že funkcia naša funkcia $f$ má vhodné vlastnosti. Môže totiž nastať prípad, že naše odhady nebudú konvergovať. Prvým prípadom, keď môže nastať problém je, že $f'(x_i) = 0$, teda narazíme na lokálny extrém funkcie $f$. Takisto problém môže nastať, ak náš počiatočný odhad $x_0$ je príliš vzdialený od skutočného koreňa $x$. Tieto problémy sa však dajú ľahko preklenúť tým,




Obr. 2.1: Príklad funkcie $e^x - 2$ s postupne sa približujúcimi aproximáciami.

že vyskúšame viacero rôznych začiatočných hodnôt $x_0$. Ani to nás nemusí ochrániť od problémov. Môže sa totiž stať, že bez ohľadu na to, ako zvolíme hodnotu $x_0$, naša postupnosť odhadov bude divergovať.

Pokúsme sa nájsť koreň funkcie $f(x) = x^{1/3}$ použitím našej metódy. Zo vzorca, ktorý sme si odvodili je jasné, že:

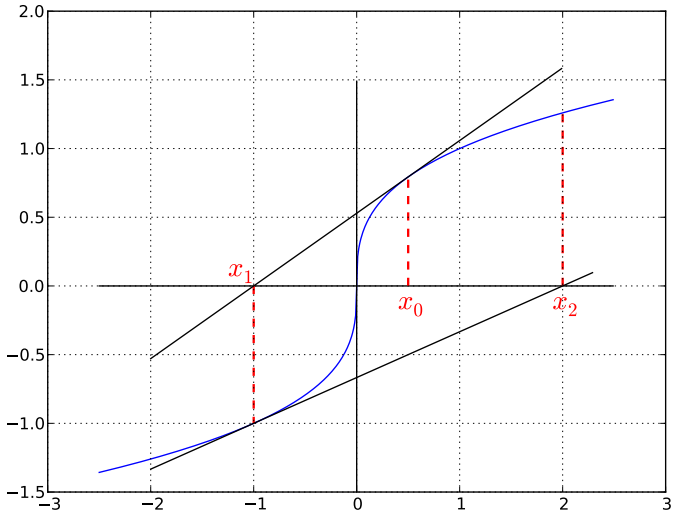
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^{1/3}}{\frac{1}{3}x_i^{-2/3}} = x_i - 3x_i = -2x_i$$

To znamená, že každou iteráciou sa hodnota $x_i$ v absolútnej hodnote zdvojnásobí a nebude sa približovať k nami očakávanej nule.

Napriek tomu je táto metóda pomerne silná a používaná. Dá sa dokonca dokázať, že naše odhady budú konvergovať ku koreňu vtedy, ak je splnená nasledovná podmienka:

$$\left| \frac{f(x)f''(x)}{f'(x)^2} < 1 \right| \tag{2.9}$$

Vidno, že pre vyššie uvedený príklad $x^{1/3}$ vychádza táto hodnota 2, takže v tomto prípade nie je konvergencia zaručená.




Obr. 2.2: Príklad funkcie $x^{1/3}$ s postupne sa vzďaľujúcimi aproximáciami.

Aby sme mohli posúdiť, nakoľko je táto metóda silná, potrebujeme sa pozrieť na to ako rýchlo konverguje k správnej hodnote. Ak sa pozrieme do malého okolia $\epsilon$ okolo koreňa $x$ funkcie $f$, tak hodnota funkcie a jej derivácie je približne:

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + \epsilon^2 \frac{f''(x)}{2} + \dots \tag{2.10}$$

$$f'(x+\epsilon) = f'(x) + \epsilon f''(x) + \dots \tag{2.11}$$

Ak si označíme $\epsilon_i$ hodnotu, od ktorej sa líši hodnota $x_i$ od skutočného koreňa $x$, z Newton-Raphsonovho vzorca vyplýva, že:

$$\epsilon_{i+1} = \epsilon_i - \frac{f(x_i)}{f'(x_i)} \tag{2.12}$$

Po dosadení prvých dvoch vyjadrení a upravení dostávame rekurentné vyjadrenie chyby $\epsilon_{i+1}$ od derivácií funckie v koreni:

$$\epsilon_{i+1} = -\epsilon^2 \frac{f''(x)}{2f'(x)} \tag{2.13}$$

To znamená, že Newton-Raphsonovou metódou konverguje náš odhad ku koreňu kvadraticky rýchlo. Teda v blízkosti koreňa sa každou iteráciou zdvojnásobí počet významných cifier odhadu.

## 2.4 Inferencia dĺžok hrán vo fylogenetickom strome

V predchádzajúcich kapitolách sme sa zoznámili so všetkými modelmi a metódami, ktoré sú potrebné na to, aby sme si vysvetlili ako vieme upravovať dĺžku hrán vo fylogenetickom strome.

Majme strom $T$ a dva významné vrcholy $a$ a $b$, medzi ktorými vedie hrana. Našim cieľom je upraviť dĺžku tejto hrany tak, aby sme nezmenili žiadnu inú hranu a zároveň náš nový strom mal čo najväčšiu pravdepodobnosť. [8, 11] Označme si dĺžku tejto hrany ako $t$. Pokúsime sa teraz vyjadriť pravdepodobnosť nášho stromu ako funkciu závisiacu od $t$.

Pozrime sa teraz na nukleotid na pozícii $u$. Priradíme vrcholu $a$ nukleotid $x$ a vrcholu $b$ nukleotid $y$. Pravdepodobnosť, že podstrom pod vrcholom $a$ vznikol z nukleotidu $x$ označíme $P(L_a|x)$. Rovnako pravdepodobnosť, že podstrom pod vrcholom $b$ vznikol z nukleotidu $y$ označíme $P(L_b|y)$. Ale tieto hodnoty sú presne tie, ktoré nám vo svojom dynamickom programovaní ráta Felsensteinov algoritmus. Obe tieto hodnoty sú nezávislé od dĺžky hrany medzi $a$ a $b$.

Potrebujeme ešte zistiť pravdepodobnosť, že sa nukleotid z $x$ zmení na nukleotid $y$. To nám však hovorí Jukes-Cantorov model a označíme túto pravdepodobnosť ako $P(x|y,t)$. Všimnime si, že táto hodnota je už závislá od dĺžky hrany $t$, lebo čas, ktorý poskytneme na mutáciu ovplyvňuje jej pravdepodobnosť.

Posledné čo zostáva je spojiť tieto hodnoty. Tie musíme postune sumovať cez všetky možné $x$ a $y$. Pravdepodobnosť nášho stromu na pozícii $u$ môžeme, v závislosti od hodnoty $t$, vyjadriť ako:

$$f_u(t) = \sum_{x \in \Sigma} \sum_{y \in \Sigma} P(x|y,t) P(L_a|x) P(L_b|y) \tag{2.14}$$

Nás ale zaujíma celková pravdepodobnosť nášho stromu. Toto je ako zvyčajne súčin cez všetky možné pozície $u$. Obrovský súčin funkcií je však zbytočne komplikovaný, namiesto toho budeme teda optimalizovať logaritmus z tejto funkcie, ktorý si označíme ako funkciu $g(t)$. Logaritmus nám totiž elegantným spôsobom odstráni súčin a nahradí ho sumou:

$$g(t) = \sum_{u} \log(f_u(t)) \tag{2.15}$$

Teraz by sme chceli nájsť takú hodnotu $t$, aby sme dostali čo najväčšiu hodnotu funkcie $g(t)$. Toto je vlastne úloha hľadania extrému v danej funkcii. Vieme však, že extrém funkcie sa nachádza v takom bode, že derivácia v tomto bode je nulová. Tu nám príde na pomoc Newton-Raphsonova metóda na hľadania koreňa funkcie. Namiesto hľadania koreňa funkcie $g(t)$, budeme hľadať koreň funkcie $g'(t)$. V koreni derivácie sa potom bude nachádzať extrém našej funkcie a teda aj naše hľadané $t$.

Na to, aby sme mohli použiť túto metódu teda potrebujeme prvú aj druhú deriváciu funkcie $g$. Pomocou nich môžeme aproximovať hodnotu $t$ Newton-Raphsonovou metódou, ako $t_{i+1} = t_i - \frac{g'(t_i)}{g''(t_i)}$ až kým hodnota $|t_{i+1} - t_i|$ nebude dostatočne malá.

# Kapitola 3

# Aplikácia na duplikačné histórie

V predchádzajúcej kapitole sme sa oboznámili s metódou, ktorá slúži na upravovanie dĺžok hrán vo fylogenetickom strome. Podobný cieľ má aj naša práca, predchádzajúci výsledok sa však nedá priamočiaro aplikovať na náš problém.

Náplňou tejto kapitoly bude teda popis toho, ako upraviť metódu na úpravu dĺžok hrán vo fylogenetickom strome, aby bola aplikovateľná na problém časovania udalostí v duplikačnej histórii. Predstavíme si model na rátanie pravdepodobnosti duplikačnej histórie, potrebné úpravy a návrh algoritmu.

## 3.1 Pravdepodobnosť duplikačnej histórie

Cieľom našej práce je vyrábať nové duplikačné histórie tým, že postupne meníme časovanie udalostí starých, už existujúcich histórií. Potrebujeme však nejaký parameter, ktorý nám bude slúžiť ako vhodné merítko toho, kedy je jeden čas udalosti lepší ako druhý čas. Najprirodzenejší spôsob je pozerať sa na to, aká je pravdepodobnosť, že náš model vygeneruje históriu s prvým časom oproti pravdepodobnosti histórie s druhým časom.

Poďme sa teda pozrieť na to, ako vyzerá náš generatívny model duplikačných histórií a ako vieme pre danú duplikačnú históriu vyrátať pravdepodobnosť jej výskytu.

### 3.1.1 Pravdepodobnostný generatívny model duplikačných histórií

Keďže je náš model pravdepodobnostný, bude dobré si na úvod spomenúť niekoľko pravdepodobnostných rozdelení, ktoré budeme používať. Najdôležitejšie z nich bude

Poissonovo pravdepodobnostné rozdelenie, ktoré úzko súvisí s exponenciálnym rozdelením. Takisto budeme používať rovnomerné a geometrické rozdelenie.

Náš model bude reprezentovaný generatívnym procesom, ktorý bude na základe daných parametrov generovať duplikačnú históriu a taktiež sekvencie zodpovedajúce jednotlivým vrcholom histórie.

Na začiatku náš model vygeneruje náhodnú sekvenciu znakov, ktorú priradí koreňu našeho stromu. Táto sekvencia predstavuje ancestrálnu DNA sekvenciu. Pre jednoduchosť budeme predpokladať, že dĺžka tejto sekvencie je daná vopred, a preto sa jej hodnota nebude rátať do celkovej pravdepodobnosti.

Postupom času môžu nastávať jednak veľké duplikačné udalosti a jednak lokálne substitúcie jednotlivých nukleotidov. Na modelovanie duplikačných udalostí budeme používať Poissonovo rozdelenie, ktorému dáme parameter $\lambda_{ler}$ (dolný index je skratka pre likelihood event rate). Tento parameter hovorí o tom, ako často sa dejú veľké udalosti.

Ak nastane veľká udalosť musíme zvoliť, ktorý typ udalosti nastal a akú časť sekvencie ovplyvnil. Ako vieme z prvej kapitoly, uvažujeme len tri typy duplikačných udalostí – duplikáciu, reverznú duplikáciu a deléciu. Pravdepodobnosť, že nastane delécia je daná hodnotou $p_d$ a pravdepodobnosť, že naša duplikácia je reverzná je $p_i$.

Ak je vygenerovaná udalosť delécia je potrebné určiť, ktorá časť sekvencie sa vymaže. Určíme si teda začiatok mazaného úseku a jeho dĺžku. Na zistenie začiatku použijeme rovnomerné rozdelenie naprieč celou dĺžkou sekvencie. Na určenie dĺžky následne použijeme geometrické rozdelenie pravdepodobnosti s priemerom $p_{ml}$. Samozrejme, takýto model môže vytvoriť deléciu, ktorá presahuje okraj sekvencie, budeme však predpokladať, že takýto prípad nenastane.

Ak vygenerujeme duplikáciu, situácia je trochu komplikovanejšia. Potrebujeme totiž určiť nie len ako vyzerá duplikovaná časť, ale takisto zistiť kam do sekvencie sa vloží duplikovaná časť. Duplikovanú časť určíme rovnako ako pri delécii. Rovnomerným rozdelením cez celú sekvenciu určíme začiatok duplikácie a geometrickým rozdelením s rovnakou pravdepodobnosťou ako pri delécii, teda $p_{ml}$, určíme dĺžku duplikovaného úseku. Následne potrebujeme určiť, kam sa táto časť naduplikuje. Musíme si však dať pozor, aby sme si nezvolili miesto vo vnútri duplikovaného úseku. Preto si radšej zvolíme vzdialenosť od tohoto úseku, a to geometrickým rozdelením s priemerom $p_{md}$. Nakoniec

už len potrebujeme určiť, či budeme kopírovať napravo alebo naľavo od zvolenej časti, táto pravdepodobosť však bude rovnomerná, teda 50 na 50. Opäť predpokladáme, že nenastane prípad, kedy by sme zasahovali mimo sekvencie.

Pomedzi toto sa dejú malé zmeny na nukleotidoch. Budeme predpokladať, že sa nedejú lokálne inzercie a delécie, ale iba substitúcie. Tie vieme modelovať Jukes-Cantorovým modelom. Z tohoto dôvodu sa ku všetkým nezrovnalostiam v zarovnaniach chováme ako ku chýbajúcim dátam.

### 3.1.2 Výpočet pravdepodobnosti duplikačnej histórie

Ak teraz na vstupe dostaneme nejakú duplikačnú históriu a takisto sekvencie, ktoré jej prislúchajú, otázkou je, s akou veľkou pravdepodobnosťou náš pravdepodobnostný model vygeneruje práve takúto históriu. Výpočet tejto pravdepodobnosti si teda rozdelíme na niekoľko častí, ktoré zodpovedajú spôsobu, ako sa naša história generuje a na konci pravdepodobnosť týchto častí vynásobíme.

Pozrime sa najskôr na samotný tvar našej duplikačnej histórie, teda na časy, kedy nastali jednotlivé udalosti. Pre každú udalosť musíme zarátať pravdepodobnosť toho, že nastane v konkrétnom čase. Keďže časy výskytu udalostí sú modelované Poissonovým rozdelením, pravdepodobnosť, že konkrétna udalosť nastala práve v danom čase po predchádzajúcej udalosti, je daná exponenciálnym rozdelením s parametrom $\lambda_{ler}$. Ak $t$ bude označovať čas od predchádzajúcej udalosti, pravdepodobnosť času zvolenej duplikačnej udalosti je $\lambda_{ler}e^{-\lambda_{ler}t}$.

Nesmieme však zabudnúť, že na poslednej hrane, ktorá končí listom, nemohla nastať žiadna ďalšia veľká udalosť. Aj túto pravdepodobnosť musíme zarátať do výsledného súčinu. Dokonca je modelovaná tým istým rozdelením. Pravdepodobnosť, že sa nič nestane $t$ jednotiek času bude teda $e^{-\lambda_{ler}t}$.

Pre každú veľkú udalosť musíme ďalej zarátať, že je daného typu a má daný tvar. Vieme, že pravdepodobnosť výskytu delécie je $p_d$ a pravdepodobnosť, že duplikácia je reverzná je $p_i$. Naviac potrebujeme zarátať pravdepodobnosť začiatku a dĺžku danej udalosti, či sa už jedná o deléciu alebo duplikáciu. Keďže začiatok má rovnomerné náhodné rozdelenie, tak výskyt na konkrétnom mieste má pravdepodobnosť $\frac{1}{len}$, kde $len$ je dĺžka celej sekvencie. Označme si dĺžku mazanej alebo duplikovanej časti $e_{len}$. Keďže táto hodnota podlieha geometrickému rozdeleniu s parametrom $p_{ml}$, pravdepodobnosť tejto konkrétnej hodnoty je $p_{ml}(1-p_{ml})^{e_{len}-1}$.

Ak je naša udalosť duplikácia, potrebujeme navyše zarátať pravdepodobnosť, že sa duplikovaný kus vyskytne na konkrétnom mieste. Označme si vzdialenosť duplikovaných častí $e_{dist}$. Keďže táto premenná je generovaná geometrickým rozdelením s parametrom $p_{md}$, dostaneme pravdepodobnosť $p_{md}(1-p_{md})^{e_{dist}}$. Naviac ešte prinásobíme $\frac{1}{2}$ za smer, v ktorom je naduplikovaná časť.

Tým sme úplne zarátali pravdepodobnosť za všetky veľké udalosti. Mimo to však nastávali substitúcie ovplyvňujúce jednotlivé nukleotidy. Pre každý atóm, ktorý sa v našej sekvencii nachádza vytvoríme, podľa danej histórie atómový strom, ktorý bude mať v listoch zarovnania zo vstupu. Pomocou Felsensteinovho algoritmu môžeme vyrátať pravdepodobnosť, že nám vznikli práve takéto zarovnania. Túto pravdepodobnosť na záver prinásobíme ku výslednej pravdepodobnosti.

## 3.2 Časovanie udalostí duplikačnej histórie

V druhej kapitole sme sa oboznámili s metódou, ktorá slúži na upravovanie dĺžok hrán vo fylogenetickom strome. Našou snahou bude upraviť túto metódu tak, aby sa dala použiť aj na duplikačné histórie a časovanie udalostí v nej.

Prístup, ktorého sa chceme držať je nasledovný. Budeme sa snažiť vyjadriť pravdepodobnosť histórie ako funkciu $f$ závisiacu od času konkrétnej duplikačnej udalosti a následne používať prvú a druhú deriváciu a Newton-Raphsonovou metódou aproximovať maximum tejto funkcie. Predpokladáme, že iteráciou tohoto postupu pre rôzne udalosti našej histórie, budeme vytvárať čoraz pravdepodobnejšie nové duplikačné histórie.

Vyberme si teda jednu konkrétnu udalosť $E$. Ako vieme, snažíme sa nájsť nový, lepší čas, ktorý nezmení poradie našich udalostí. Čas kedy mohla udalosť $E$ nastať je teda ohraničený časom predchádzajúcej a nasledujúcej udalosti. Ak sa však pozrieme do nášeho pravdepodobnostného modelu, zistíme, že všetky pravdepodobnosti, ktoré sú závislé od časov udalostí nevyžadujú absolútny čas udalosti, ale relatívny čas od susedných udalostí. Premenná našej funkcie $t$ bude teda určovať o koľko skôr nastala udalosť $E$ oproti nasledujúcej udalosti. Premenná $t$ teda leží v intervale od 0 po hodnotu $maxtime$. Hodnotu $maxtime$ určíme ako rozdiel časov predchádzajúcej a nasledujúcej udalosti, alebo ekvivalentne, ako súčet časov dvoch hrán, ktoré sú incidentné s udalosťou $E$.

Poďme sa teraz pozrieť na to, ako vyzerá pravdepodobnosť našej histórie v závislosti od hodnoty $t$. Dobrou novinkou je, že môžeme absolútne zanedbať pravdepodobnosť, ktorá je ovplyvnená tvarom udalostí. Nemusíme sa teda pozerať na to, kde v našej sekvencii nastala duplikácia či delécia a ani to, aká bola dlhá alebo ktorým smerom išla. Tieto vlastnosti totiž nie sú ovplyvnené časom jednotlivých udalostí, a keďže my nemeníme tvar udalostí, táto pravdepodobnosť je vždy rovnaká bez ohľadu na načasovanie udalosti $E$.

V rámci veľkých udalostí musíme zarátať iba pravdepodobnosť toho, že sa táto udalosť vyskytne práve $maxtime - t$ jednotiek času po predchádzajúcej udalosti, a že sa ďalšia udalosť udeje $t$ jednotiek po nej. Všimnime si, že nás takisto nezaujíma pravdepodobnosť, ktorá je ovplyvnená zvyšnými hranami, dĺžku týchto hrán totiž tiež nemeníme. Do našej výslednej funkcie $f$ sa teda prinásobí člen $\lambda_{ler}e^{-\lambda_{ler}(maxtime-t)}$ a člen $\lambda_{ler}e^{-\lambda_{ler}t}$ alebo $e^{-\lambda_{ler}t}$ v závislosti od toho, či je udalosť $E$ posledná udalosť v našej histórii.

Ak sa však na to pozrieme lepšie, uvedomíme si, že aj táto pravdepodobnosť je konštantná. Po vynásobení týchto dvoch členov totiž dostaneme $c_1e^{-\lambda_{ler}(maxtime-t)}e^{-\lambda_{ler}t}$ pre nejakú konštantu $c_1$, čo vieme upraviť na $c_1e^{-\lambda_{ler}maxtime}$. Vypadla nám teda premenná $t$ a tento člen pravdepodobnosti je tiež konštantný.

Problém však je, že načasovaním udalosti $E$ sú ovplyvnené aj niektoré stromy atómov, ktoré výrazne ovplyvňujú pravdepodobnosť histórie. Opäť si treba uvedomiť, že ak nejaký atóm nebol ovplyvnený udalosťou $E$, jeho atómový strom sa nemení. V atómovom strome totiž udalosť $E$ nevytvorí žiaden deliaci vrchol. Na danom mieste vytvorí len cestu a z multiplikatívnosti Jukes-Cantorovho modelu je jasné, že táto cesta bude mať bez ohľadu na hodnotu $t$ stále rovnakú pravdepodobnosť, lebo $S(t) + S(maxtime - t) = S(maxtime)$.

Zaujímajú nás teda len tie atómové stromy, ktorých zodpovedajúci atóm bol ovplyvnený udalosťou $E$. Pravdepodobnosť atómového stromu rátame Felsensteinovým algoritmom. Pozrime sa teda na to, ako nám hodnota $t$ ovplyvňuje túto pravdepodobnosť. Ako sme si už spomínali, tento algoritmus zakaždým ráta pravdepodobnosti podstromov pod nejakým vrcholom $P(L_\ell|a)$ a túto pravdepodobnosť ráta pomocou vzťahu:

$$P(L_\ell|a) = \sum_{b,c\in\Sigma} P(b|a,t_i)P(L_i|b)P(c|a,t_j)P(L_j|b) \tag{3.1}$$

Všetko nižšie, ako vrcholy zodpovedajúce udalosti $E$, má túto hodnotu konštantnú, nezávislú od hodnoty $t$. Prvýkrát teda premenná $t$ vstupuje do našej pravdepodobnosti

až keď sa dostaneme do takého vrchola, z ktorého vedie do synov hrana dĺžky $t$.

Keďže hodnoty $P(L_i|b)$ a $P(L_j|c)$ sú konštanty, zaujímajú nás len zvyšné dve. Tie sú ale z Jukes-Cantorovho modelu buď rovné $\frac{1}{4}(1+3e^{-4\alpha t})$ alebo $\frac{1}{4}(1-e^{-4\alpha t})$, v závislosti od toho, či sa $a$ rovná alebo nerovná $b$ a $c$. Uvedomme si, že tieto pravdepodobnosti majú pekný tvar

$$c_1 e^{-4\alpha 0 t} + c_2 e^{-4\alpha 1 t} \tag{3.2}$$

pre nejaké konštanty $c_1$ a $c_2$.

Hodnota $t$ sa však prejaví ešte na jednom mieste, a to na hrane, ktorá vedie do otca vrcholu vytvoreného udalosťou $E$. Táto hrana má totiž dĺžku $maxtime - t$. Ak sa pozrieme na pravdepodobnosti $P(b|a, maxtime - t)$ zistíme, že je rovná buď $\frac{1}{4}(1+3e^{-4\alpha(maxtime-t)})$ alebo $\frac{1}{4}(1-e^{-4\alpha(maxtime-t)})$. To môžeme opäť upraviť na tvar $c_1 e^{-4\alpha 0(maxtime-t)} + c_2 e^{-4\alpha 1(maxtime-t)}$.

Výraz $e^{-4\alpha 1(maxtime-t)}$ však vieme upraviť na $e^{-4\alpha maxtime} e^{-4\alpha -1t}$, pričom prvá mocnina neobsahuje premennú $t$. Môžeme ju teda chápať ako konštantu. Vidíme, že takáto pravdepodobnosť sa dá opäť upraviť na tvar

$$c_1 e^{-4\alpha 0 t} + c_2 e^{-4\alpha -1 t} \tag{3.3}$$

Pre jednoduchosť si označme hodnotu $-4\alpha$ ako $c$. Vidíme, že naše pravdepodobnosti obsahujú iba členy $c_i e^{cit}$ pre nejaké $i$. Dokonca aj ľubovoľná konštanta vie byť reprezentovaná takouto formou, ak nastavíme $i = 0$. Nastolíme teda hypotézu, že všetky pravdepodobnosti, ktoré sa vyskytujú vo Felsensteinovom algoritme sa dajú zapísať ako $\sum_{i\in\mathbb{Z}} c_i e^{cit}$. Pre jednoduchosť môžeme ako indexovú množinu brať celé $\mathbb{Z}$, pre členy, ktoré tam nie sú prítomné bude hodnota $c_i = 0$.

Skúsme túto hypotézu dokázať. Ako sme už spomínali, všetky elementárne pravdepodobnosti, či už konštantné alebo závislé od $t$, majú takýto tvar. Potrebujeme teda ukázať, že všetky operácie, ktoré s týmito funkciami Felsensteinov algoritmus vykonáva, zachovávajú ich tvar. Nie je ťažké si všimnúť, že vo Felsensteinovom algoritme používam len sčítanie a násobenie.

Pozrime sa najskôr na sčítanie. A pre jednoduchosť pričítavajme len jeden člen, $d_j e^{cjt}$. Potom vidíme, že ak takýto člen pripočítame k našej funkcii tvaru $\sum_{i\in\mathbb{Z}} c_i e^{cit}$ dostaneme výraz $\sum_{i<j} c_i e^{cit} + (c_j + d_j) e^{cjt} + \sum_{i>j} c_i e^{cit}$. Tento výraz má však požadovaný tvar, keďže $c_j + d_j$ je opäť iba konštanta. Vidíme teda, že pričítanie jedného takého

člena nič nezmení a súčet dvoch funkcií takéhoto tvaru nie je nič iné ako postupné pričítavanie členov jednej funkcie k druhej. Keďže sa zakaždým zachováva tvar, na konci tiež dostávame funkciu správneho tvaru a takáto trieda funkcií je teda uzatvorená na súčet.

Takým istým spôsobom môžeme pristúpiť aj k násobeniu. Nech chceme prinásobiť člen $d_j e^{cjt}$ k funkcii nášeho tvaru. Výsledná funkcia vyzerá nasledovne: $\sum_{i\in\mathbb{Z}} c_i d_j e^{c(i+j)t}$. Takýto tvar je však opäť vyhovujúci. No a súčin dvoch funkcií nášeho tvaru je len postupné vynásobenie druhej funkcie členmi prvej funkcie a potom sčítanie výsledných funkcií. Keďže všetky tieto kroky zachovávajú tvar funkcie, aj výsledná funkcia má správny tvar.

To znamená, že všetky pravdepodobnosti, ktoré ráta Felsensteinov algoritmus, majú pekný tvar $\sum_{i\in\mathbb{Z}} c_i e^{cit}$. Keď rátam pravdepodobnosť atómového stromu, Felsensteinov algoritmus počíta pravdepodobnosti pre jednotlivé nukleotidy. Výsledné pravdepodobnosti sa následne násobia dokopy, to nám však stále nenarúša tvar funkcie. A takisto vynásobenie pravdepodobností pre všetky atómové stromy nám dá len funkciu takéhoto tvaru.

Všimnime si, že na rozdiel od fylogenetického stromu, nebudeme hľadať maximum logaritmu tejto funkcie. Dôvod na to je ten, že pri úprave hrany fylogenetického stromu sme dostali iba jeden člen s konštantou a logaritmovanie preto bolo oveľa jednoduchšie. V tomto prípade by niečo také bolo skoro nemožné.

Dôvod, prečo je veľmi príhodné, že pravdepodobnosti atómových stromov sú takéhoto tvaru je ten, že sa veľmi dobre derivujú v závislosti od času $t$.

$$\left(\sum_{i\in\mathbb{Z}} c_i e^{cit}\right)' = \sum_{i\in\mathbb{Z}} c_i c i e^{cit} \tag{3.4}$$

$$\left(\sum_{i\in\mathbb{Z}} c_i e^{cit}\right)'' = \sum_{i\in\mathbb{Z}} c_i c^2 i^2 e^{cit} \tag{3.5}$$

Ukázali sme si, že pomocou Felsensteinovho algoritmu vieme efektívne zisťovať pravdepodobnosť atómových stromov v závislosti od času $t$ udalosti $E$. Všetky tieto pravdepodobnosti sú dokonca jednoducho derivovateľné pomocou vzorcov 3.4 a 3.5. Ak si teda označíme $f$ funkciu, ktorá vznikne ako súčin pravdepodobností pre jednotlivé atómové stromy, nič nám nebráni použiť Newton-Rapsonovu metódu na hľadanie maxima tejto funkcie.

# Kapitola 4

# Implementácia a experimenty

V predchádzajúcich kapitolách sme si ukázali teoretický popis riešenia časovania udalostí v duplikačných históriách. V tejto kapitole si povieme niečo o našej implementácii tohoto riešenia a ukážeme niekoľko experimentov, ktoré nám pomôžu vyhodnotiť úspešnosť našej metódy.

## 4.1 Implementácia

Počas implementácie našeho riešenia sa vyskytlo niekoľko problémov, ktoré boli čisto implementačného charakteru. Postupom času sa nám podarilo všetky úspešne vyriešiť. Jeden z nich bol však kľúčový pre celý algoritmus, a preto si popíšeme tento problém aj jeho riešenie podrobnejšie.

Dôležitou časťou bolo rozhodnúť sa, ako si budem pamätať funkcie vyjadrujúce pravdepodobnosť v závislosti od hodnoty $t$. Vieme, že naša funkcia má tvar $\sum_{i\in\mathbb{Z}} c_i e^{-4\alpha i t}$, pričom funkcie, ktoré sa vyskytujú v našom programe, majú členy $i$ z nejakého súvislého rozsahu, ktorý obsahuje nulu. Existujú teda dve čísla $\ell$ a $u$ také, že naša funkcia je tvaru $\sum_{\ell\leq i\leq u} c_i e^{-4\alpha i t}$. Naviac jediné, čo sa mení člen od členu, je hodnota konštanty $c_i$. Preto sme sa rozhodli pamätať si len hodnotu $\ell$ a pole čísiel $c_l$ až $c_u$.

Tu však nastal najväčší problém. Čísla, s ktorými pracujeme, vyjadrujú pravdepodobnosť, sú to teda reálne čísla menšie ako 1, pričom tieto čísla veľakrát spolu násobíme. Problém však je, že tieto čísla sa postupne čoraz viac zmenšujú až sú menšie ako presnosť premennej *double* v C++.

Potrebovali sme teda spôsob, ako si reprezentovať malé reálne čísla. Bežne používaným prístupom je zlogaritmovanie týchto čísel. Namiesto 0.04 si teda zapamätáme

číslo $\ln(0.04) = -3.218876$. Takýto prístup nám umožňuje naozaj veľkú presnosť. Uvedomme si, že najväčšia možná pravdepodobnosť je 1, čoho logaritmus je číslo 0. A ako uvidíme v testoch na dátach, pravdepodobnosti našich histórií budú dávať po zlogaritmovaní čísla od niekoľko mínus stoviek po niekoľko mínus tisícok. A číslo $e^{-1000}$ je približne $5 \cdot 10^{-435}$. Takúto presnosť by sme určite nevedeli dosiahnuť so žiadnou pôvodnou reálnou premennou. Pamätať si však záporné čísla do niekoľkých tisíc nie je žiaden problém.

Tento prístup bohužiaľ nepokrýva všetko čo potrebujeme. Čísla, s ktorými pracujeme totiž môžu byť aj záporné. Samozrejme, žiadna pravdepodobnosť nebude záporné reálne číslo, ale v medzivýsledkoch a v koeficientoch našej funkcie sa môžu objavovať aj takéto hodnoty. A problémom je, že pre záporné čísla nie je logaritmus definovaný. Rozhodli sme sa preto, že si vytvoríme triedu $Number$, ktorá bude reprezentovať čísla, s ktorými pracujeme pomocou logaritmu absolútnej hodnoty pôvodného čísla a znamienka.

### Listing programu (C++)

```
class Number {
    double data;
    int zap;
};

Number::Number(double value) {
    data = log(abs(value));
    if(value < 0) zap = 1;
    else zap = 0;
}
```

Musíme si však uvedomiť, že tieto hodnoty nemôžeme nikde v programe prevádzať späť na pôvodné čísla tým, že by sme ich umocnili. Výsledné čísla by boli totiž tak malé, že by sme stratili akúkoľvek presnosť. Všetky matematické operácie, teda musíme vykonávať priamo v zlogaritmovanom tvare.

Nech máme dve reálne čísla $a$ a $b$ a ich logaritmy si označme $a'$ a $b'$. Ak potrebujeme zistiť ich súčin, situácia je pomerne jednoduchá, pretože $\ln(ab) = \ln(a) + \ln(b) = a' + b'$. Ak teda potrebujem násobiť dve reálne čísla, viem túto operáciu nahradiť sčítaním ich zlogaritmovaných hodnôt a dostanem zlogaritmovanú hodnotu ich súčinu.

### Listing programu (C++)

```
Number operator*(const Number& n1, const Number& n2) {
    if(n1.zero()) return Number(0.0);
    if(n2.zero()) return Number(0.0);
    Number res = Number();
    res.data = n1.data + n2.data;
    res.zap = n1.zap * n2.zap;
```

```
    return res;
}
```

Najväčšie problémy nám však spôsobila operácia sčítania. Neexistuje totiž identita, ktorá by upravovala $\ln(a+b)$ do tvaru, ktorý obsahuje iba členy $\ln a$ a $\ln b$. Predpokladajme, že číslo $b$ je väčšie ako číslo $a$. Výraz $\ln(a+b)$ si môžeme zapísať ako $\ln(b(1+\frac{a}{b}))$. Pomocou predtým spomenutej identity to vieme upraviť na $\ln(b) + \ln(1 + \frac{a}{b})$. Ďalej vieme, že $\ln a = a'$ a $\ln b = b'$. To je však ekvivalentné s tým, že $a = e^{a'}$ a $b = e^{b'}$. Náš výraz dostane tvar $b' + \ln(1 + e^{a'-b'})$. Tento tvar už obsahuje iba zlogaritmované tvary našich čísiel. Zistili sme teda, že

$$\ln(a + b) = b' + \ln(1 + e^{a'-b'}) \tag{4.1}$$

### Listing programu (C++)

```
Number operator+(const Number& n1, const Number& n2) {
    if(n1.zero()) return n2;
    if(n2.zero()) return n1;
    if(n1.zap == n2.zap) {
        Number res = Number();
        res.zap = n1.zap;
        if(n1.data < n2.data) res.data = n2.data + log(1 + exp(n1.data - n2.data));
        else res.data = n1.data + log(1 + exp(n2.data - n1.data));
        return res;
    } else {
        Number res = Number();
        if(n1.data < n2.data) {
            res.zap = n2.zap;
            res.data = n2.data + log(1 - exp(n1.data - n2.data));
        } else {
            res.zap = n1.zap;
            res.data = n1.data + log(1 - exp(n2.data - n1.data));
        }
    }
}
```

Vidíme, že naša trieda *Number* je prijateľným reprezentovaním čísiel v našom programe, keďže podporuje všetky funkcie, ktoré vyžadujeme a zároveň nám dovoľuje vyjadriť veľmi malé čísla, s ktorými by sme inak nemohli pracovať.

Čo sa týka implementácie, chcel by som ešte spomenúť časovú zložitosť navrhovaného riešenia. Síce sme ju nikdy nevyjadrovali presne a ani nás tento parameter natoľko nezaujíma, je dobré sa však zamyslieť nad tým, s akými veľkými štruktúrami budeme pracovať. Najviac nás zaujímajú funkcie, ktoré vytváram. Ak by počet ich členov rástol napríklad exponencionálne od počtu nukleotidov, naše riešenie by určite nebolo použiteľné na reálnych dátach, ktoré obsahujú státisíce nukleotidov.

To našťastie nie je náš prípad. Treba si uvedomiť, že ak násobím dve funkcie, kde v jednej si pamätám rádovo $x$ a v druhej rádovo $y$ členov, výsledná postupnosť bude

mať rádovo $x + y$ členov. A každá hrana, ktorú budeme v našom programe spracovávať nám pridá najviac funkciu s dvoma členmi, aj to len v prípade, že dĺžka tejto hrany je závislá od $t$. A keďže pracujeme na strome, počet hrán, ktoré spracúvame je lineárne závislý od počtu nukleotidov, teda dĺžky sekvencie. Z toho vyplýva, že počet členov našej funkcie bude rádovo rovný dĺžke sekvencie na vstupu. Takýto výsledok je teda pomerne prijateľný a časovo zvládnutľný.

## 4.2 Experimenty

Účinnosť našej metódy sme vyhodnotili pomocou niekoľkých sád simulovaných dát, na ktorých sme spustili náš program. Tieto dáta sa dali rozdeliť do dvoch kategórií v závislosti od veľkosti dát. Malé dáta obsahovali histórie, ktoré mali jednu až štyri evolučné udalosti. Naviac atómy obsahovali desiatky až stovky nukleotidov. Veľké dáta obsahovali tri až trinásť evolučných udalostí a atómy dĺžky až niekoľko tisícok nukleotidov. Tieto dáta sa veľkosťou podobajú skutočným dátam.

Kvôli rozličnej veľkosti dát a ich náročnosti na testovanie, sme k testovaniu oboch druhov dát pristupovali odlišne. V nasledujúcich podkapitolách si teda popíšeme postupy, ktoré sme použili a výsledky, ktoré sme vďaka nim dosiahli.

### 4.2.1 Malé dáta

Tým, že tieto dáta boli také malé, mohli sme pustiť veľa testov za sebou a postupne našu históriu čoraz viac zlepšovať. Na testovanie týchto dát sme použili nasledovný postup. Vybrali sme si náhodnú udalosť, ktorá sa v nej nachádzala a pomocou prístupu spomenutého v predchádzajúcej kapitole sme sa snažili nájsť lepší čas pre túto udalosť. Ak sa nám podarilo touto zmenou zlepšiť pravdepodobnosť histórie, túto zmenu sme si zapamätali a históriu podľa nej príslušne upravili. Ak však nový čas vytvoril horšiu históriu, túto zmenu sme ignorovali. Nasledovnú procedúru sme opakovali 100 krát, čím sme čoraz viac zlepšovali našu históriu.

Dáta, ktoré sme mali na vstupe obsahovali dva typy histórií. Na základe udalostí boli obe identické. Líšili sa len ich načasovaním. V prvej histórii boli udalosti od seba vzdialené v rovnomerných rozostupoch. Druhá história bola kontrolná a predstavovala skutočnú históriu, podľa ktorej boli vygenerované dáta. V testoch sme pustili náš program na histórii s rovnomernými časmi a porovnávanie sme robili aj s kontrolnou históriou. Výsledky týchto testov si môžete prezrieť v nasledujúcej tabuľke.

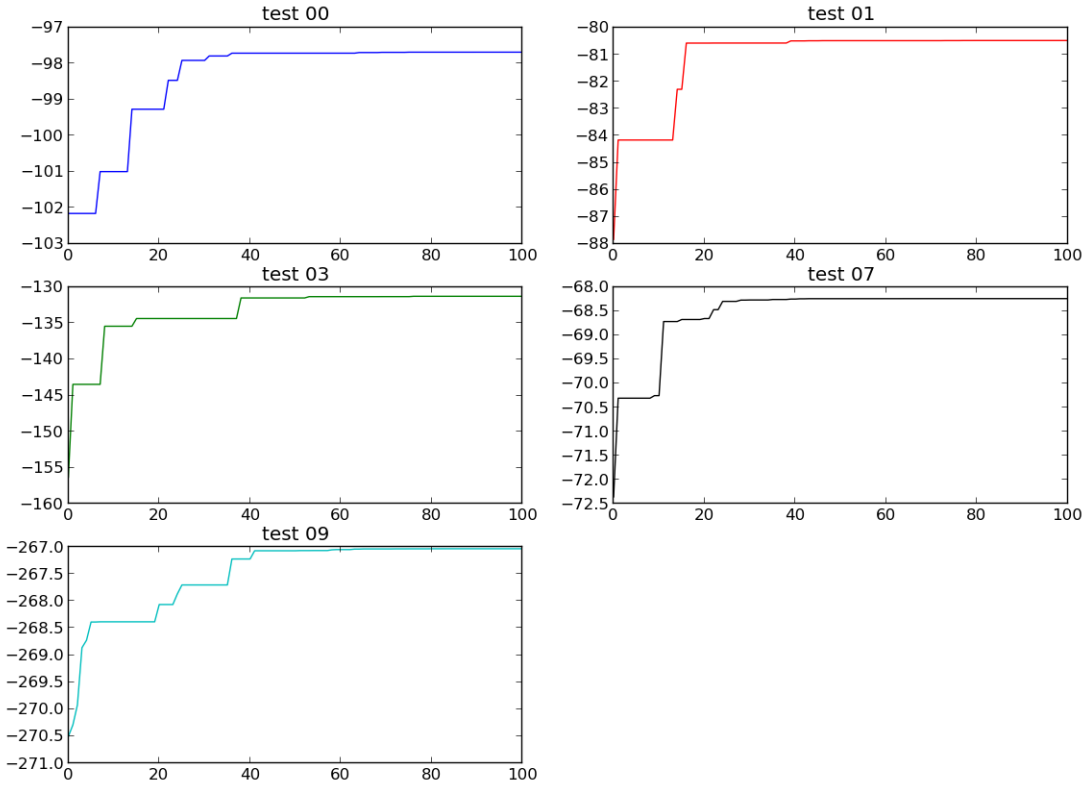
| id testu | počet udalostí | logaritmus pravdepodobnosti kontrolnej histórie | logaritmus pravdepodobnosti rovnomernej histórie | logaritmus pravdepodobnosti dosiahnutej histórie |
|---|---|---|---|---|
| 00 | 3 | −99.9897 | −102.1530 | −97.6810 |
| 01 | 2 | −88.3307 | −87.8305 | −80.4778 |
| 02 | 1 | −50.8916 | −50.8916 | −50.8916 |
| 03 | 3 | −147.9863 | −156.3289 | −131.2486 |
| 04 | 1 | −38.5977 | −38.5977 | −38.5977 |
| 05 | 1 | −36.9902 | −36.9902 | −36.9902 |
| 06 | 1 | −42.3098 | −42.3098 | −42.3098 |
| 07 | 2 | −72.6758 | −72.3582 | −68.2356 |
| 08 | 1 | −41.9665 | −41.9665 | −41.9665 |
| 09 | 4 | −270.1846 | −270.5045 | −267.0330 |

Tabuľka 4.1: Tabuľka dosiahnutých pravdepodobností

Vidíme, že ak bol počet udalostí v našej histórii rovný 1, nepodarilo sa nám zlepšiť pravdepodobnosť tejto histórie. Vo všetkých ostatných prípadoch sa nám však poradilo zlepšiť našu históriu, dokonca sme dosiahli lepšiu pravdepodobnosť ako mala naša kontrolná história. Uvedomme si, že zlepšenie oproti kontrolnej histórii nie je zlým znamením. Aj keď podľa tejto histórie boli vygenerované dáta, neznamená to, že táto história je najpravdepodobnejšia vzhľadom na dané dáta. Dokonca vidno, že ani nie je najpravdepodobnejšou, keďže sa nám ju podarilo prekonať.

Poďme si naše experimenty rozobrať podrobnejšie. Časť nášho riešenia je založená na náhode, či už pri vyberaní udalosti, ktorú budeme upravovať, alebo pri výbere začiatočného koreňa v Newton-Raphsonovej metóde. Mohlo by nás preto zaujímať, koľko takýchto náhodných iterácií potrebujeme k tomu, aby sme dokonvergovali k dosiahnutému riešeniu a teda či sa dá predpokladať, že vieme dosiahnuť aj lepšie riešenie.

Ako vidíme podľa obrázka 4.1 zhruba po 40 iteráciách sme už mali záverečné riešenie, alebo sme boli veľmi blízko. Môžeme teda predpokladať, že pridanie počtu iterácií na daných vstupných dátach by nebolo prospešné a nami dosiahnuté hodnoty pravdepobobne tvoria lokálne maximum.




Obr. 4.1: Grafy závislosti logaritmu dosiahnutej pravdepodobnosti (os y) od počtu iterácií nášeho algoritmu (os x)

### 4.2.2 Veľké dáta

Vzhľadom na veľkosť dát, nebolo možné použiť ten istý prístup, ako v predchádzajúcej časti. Dá sa totiž predpokladať, že počet iterácií potrebných na to, aby sme sa dostali k nejakému lokálnemu maximu bude väčší ako pri malých dátach. Rozhodli sme sa preto zistiť, či náš algoritmus nájde aspoň nejakú lepšiu históriu. Na každom z desiatich veľkých vstupoch sme teda pustili 10 krát náš algoritmus a sledovali, či sa aspoň v jednom z týchto behov zlepší vstupná história. Opäť to porovnávame s kontrolnou históriou. Výsledky z tohoto merania sú v nasledujúcej tabuľke:

| id testu | počet udalostí | logaritmus pravdepodobnosti kontrolnej histórie | logaritmus pravdepodobnosti rovnomernej histórie | logaritmus pravdepodobnosti dosiahnutej lepšej histórie |
|---|---|---|---|---|
| 00 | 10 | −12744.0372 | −12802.1764 | −12801.4430 |
| 01 | 3 | −3441.5563 | −3436.6462 | −3430.4116 |
| 02 | 13 | −41487.7023 | −42639.1088 | −42637.6234 |
| 03 | 10 | −23483.1279 | −22937.3864 | −22932.1529 |
| 04 | 11 | −36454.2854 | −36497.9842 | −36492.7121 |
| 05 | 14 | −17071.6720 | −17070.5347 | −17053.2624 |
| 06 | 11 | −13418.8370 | −12561.3603 | nenájdená |
| 07 | 9 | −19787.6913 | −20103.9035 | nenájdená |
| 08 | 7 | −24849.3075 | −24901.2296 | −24894.4932 |
| 09 | 7 | −7383.3925 | −7726.4870 | −7712.3483 |

Tabuľka 4.2: Tabuľka dosiahnutých pravdepodobností

Vidíme, že vo väčšine testovacích vstupov sa nám podarilo na desať iterácií zlepšiť históriu, ktorú sme mali na vstupe. To naznačuje, že počet iterácií, ktoré potrebujeme na nájdenie nejakého lokálneho maxima by nemusel byť až taký veľký a náš algoritmus by s dostatkom času mohol produkovať ešte lepšie histórie.

# Záver

V tejto práci sme sa zaoberali problematikou časovania udalostí v rámci duplikačných histórií. Inšpirovali sme sa prístupom, ktorý slúži na upravovanie dĺžok hrán vo fylogenetickom strome, pričom sa nám ho podarilo prispôsobiť a aplikovať na náš problém časovania udalostí v duplikačných históriách. Naše riešenie sme implementovali a výsledný program sme následne otestovali na simulovaných dátach s pozitívnym výsledkom, kde sa nám podarilo zlepšiť histórie zo vstupu a pravdepodobne dosiahnuť lokálne maximum.

Ukazuje sa, že takýto prístup môže byť veľmi prínosný pri úprave duplikačných histórií a môže byť použitý ako doplňujúci nástroj k programom, ktoré vytvárajú duplikačné histórie, akým je napríklad práca Jána Hozzu [5].

Samozrejme, náš prístup je ešte v začiatočnej fáze a má potenciál na zlepšenia. Bolo by potrebné spraviť dôkladnejšie testy na veľkých dátach a pozrieť sa, kedy sa naša história zlepšuje a aký tvar má výsledná história. Takisto sa dá zlepšovať samotná implementácia, či už optimalizáciou niektorých procedúr za cieľom zníženia časovej náročnosti, alebo skúšaním iných numerických a heuristických metód na hľadanie maxima funkcie.

Netreba tiež zabudnúť, že nás prístup upravuje históriu len veľmi lokálne pomocou časovanie jednej udalosti. Stálo by za zváženie preskúmať prístupy, ktoré hýbu naraz viacerými udalosťami.

# Literatúra


[1] G.L. Bradley and K.J. Smith. *Calculus*. Prentice Hall, 1995.

[2] 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.

[3] Richard Durbin, Sean Eddy, Anders Krogh, and Graeme Mitchison. Biological sequence analysis: probabilistic models of proteins and nucleic acids. cambridge univ, 1998.

[4] Exelixis Lab. Pll - phylogenetic likelihood library, 2014. http://www.libpll.org/.

[5] Ján Hozza. Rekonštrukcia duplikačných histórií pomocou pravdepodobnostného modelu. Bakalárska práca, Univerzita Komenského v Bratislave, 2014.

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

[7] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[8] A. Stamatakis, T. Ludwig, and H. Meier. RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics*, 21(4):456–463, 2005.

[9] 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.

[10] 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.

[11] Z. Yang. Maximum likelihood estimation on large phylogenies and analysis of adaptive evolution in human influenza virus A. *J Mol Evol*, 51(5):423–432, 2000.