Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Rekonštrukcia histórií génových zhlukov

Diplomová práca

2016

Bc. Ján Hozza

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Rekonštrukcia histórií génových zhlukov

Diplomová práca

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

Bratislava, 2016
Bc. Ján Hozza

97411022

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Ján Hozza
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Rekonštrukcia histórií génových zhlukov
Reconstruction of gene cluster histories

Cieľ: Cieľom práce je navrhnúť, implementovať a vyhodnotiť nový prístup na rekonštrukciu evolučných histórií génových zhlukov použitím optimalizačných metód ako napríklad simulované žíhanie. Z informatického hľadiska sú génové zhluky reťazce, ktoré sa menia v čase kopírovaním podreťazcov na blízke pozície. Takéto udalosti veľkého rozsahu sú doplnené náhodnými lokálnymi zmenami v reťazci. Študent navrhne skórovaciu funkciu inšpirovanú pravdepodobnostným modelom tohto procesu a vytvorí efektívny prístup na získanie rekonštrukcií s vysokým skóre.

Vedúci: Mgr. Tomáš Vinař, PhD.
Rektorát, dekanát: FMFI.Dek - Dekanát

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 14.12.2014

Dátum schválenia: 17.12.2014 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie

Chcel by som poďakovať môjmu vedúcemu Tomášovi Vinařovi, za trpezlivosť, cenné rady, obetovaný čas. Tiež by som chcel poďakovať svojim rodičom a priateľom najmä za psychickú podporu.

Abstrakt

Keď nastane viacero dlhých duplikácii v jednej časti DNA sekvencie, môže to mať za následok vznik génového zhluku so zložitou štruktúrou. Aby sme mohli lepšie porozumieť funkcii génov v zhluku, snažíme sa odhaliť, aké dlhé mutácie v sekvencii nastali. Tento problém sa nazýva rekonštrukcia histórie sekvencie.

V tejto práci vylepšíme predošlý algoritmus na rekonštrukciu histórií využitím informácií získaných z evolučných stromov pre úseky sekvencie. Pomocou metód strojového učenia naučíme algoritmus robiť komplexné rozhodnutia a rozlišovať medzi správnymi a nesprávnymi duplikáciami.

Podrobne preskúmame úspešnosť nových metód a porovnáme rôzne varianty algoritmu.

Kľúčové slová: DNA sekvencia, génový zhluk, rekonštrukcia, strojové učenie, evolúcia

Abstract

A series of long duplications in a specific region of DNA sequence can lead to complex-structured gene clusters. It is easier to understand the purpose of genes in the cluster, if we untangle the structures by identifying what long mutations created the cluster.

We used evolution trees of certain parts of the DNA sequence to improve the previous algorithm for the reconstruction of histories. We further used machine learning to teach algorithm to make difficult choices and distinguish between correct and incorrect duplications.

We have evaluated the impact of our improvements on reconstruction using different variants of our algorithm.

Keywords: DNA sequence, gene cluster, reconstruction, machine learning, evolution

Obsah

Úvod **1**

1 Základné pojmy **2**
1.1 DNA sekvencie 2
1.2 Mutácie v DNA sekvenciách 2
1.3 Génové zhluky a ich analýza 3
1.4 História DNA sekvencie 4
1.5 Atómy 4
1.6 Duplikačná udalosť 5

2 Algoritmus na rekonštrukciu histórií **6**
2.1 Problém rekonštrukcie histórie 6
2.2 Rozdelenie sekvencie na atómy 7
2.3 Schéma algoritmu 7
2.4 Výber poslednej udalosti 8
2.5 Porovnávanie udalostí a histórií 9

3 Generovanie kandidátov na udalosti **12**
3.1 Algoritmus na rýchle vzorkovanie kandidátov 12
3.2 Využitie DNA sekvencií atómov pri rekonštrukciu histórií 15
3.3 Evolučné stromy atómov 16
3.4 Využitie evolučných stromov atómov 18
3.5 Čerešňovitosť 20
3.6 Využitie čerešňovitosti pri návrhu kandidátov 22
3.7 Dôvera v čerešňovitosť 22

4 Skórovanie udalostí **24**
4.1 Susedné páry atómov a predĺžiteľnosť udalostí 24
4.2 Zoznam parametrov udalostí 27
4.3 Výpočet celkového skóre 29

4.4 Logistická regresia 31
4.5 Generovanie simulovaných histórií 32

5 Experimentálne výsledky **34**
5.1 Doladenie parametrov 34
5.2 Charakteristiky simulovaných histórií 34
5.3 Generovanie kandidátov 35
5.4 Trénovanie modelu logistickej regresie 37
5.5 Skórovanie udalostí 39
5.6 Rekonštrukcia histórií 41

Záver **43**

Úvod

Vlastnosti všetkých mnohobunkových organizmov sú zakódované do DNA sekvencií, ktoré uchovávajú všetky potrebné informácie pre život bunky ale aj celého organizmu. V týchto DNA sekvenciách sa dejú rôzne zmeny, medzi ktoré patria aj duplikácie dlhých úsekov DNA. Viacero takýchto duplikácii v jednej časti DNA môže mať za následok vznik zložitých génových zhlukov.

Rozpliesť štruktúru zhlukov a zistiť, aké duplikácie viedli ku ich vzniku je ťažký problém a nemôžeme na jeho riešenie použiť štandardné metódy aplikovateľné na úseky DNA s malým počtom duplikácií.

Na riešenie tohto problému použijeme pravdepodobnostný algoritmus, ktorý bude generovať postupnosť histórií, ktoré by viedli skúmanej DNA sekvencii zhluku, pričom pre každú históriu budeme postupne rekonštruovať udalosti od tých, ktoré nastali naposledy až tie, ktoré zmenili DNA sekvenciu pred desiatkami miliónov rokov.

Sústrediť sa budeme na to, aby sa vo výslednej sekvencii nachádzala aj správna história, pričom v tejto práci sa nezaoberáme tým, ako napokon zistiť, ktorá z nich to je.

V prvej kapitole zavedieme a vysvetlíme základné pojmy, ktoré budeme ďalej v práci používať. Povieme si napríklad, čo je to história DNA sekvencie a čo to znamená rozdeliť sekvenciu na atómy.

V druhej kapitole si zadefinujeme problém rekonštrukcie a predstavíme si algoritmus, ktorý bude problém riešiť. Tento algoritmus má dve zložitejšie časti, ktoré sú však veľmi dôležité pre celkovú úspešnosť algoritmu.

V tretej kapitole ukážeme, ako je možné využiť evolučné stromy atómov na vylepšenie algoritmu, ktorý vzorkuje kandidátov na udalosti.

V štvrtej kapitole popíšeme duplikačné udalosti veľkým množstvom parametrov a následne ukážeme na základe týchto parametrov rozhodovať, ktoré udalosti sú správne.

V piatej kapitole podrobíme algoritmus dôkladným testom a preskúmame, ako rôzne prístupy ku riešeniam jednotlivých problémov ovplyvňujú úspešnosť rekonštrukcie.

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

Kapitola 1

Základné pojmy

1.1 DNA sekvencie

Pre účely tejto práce bude *DNA sekvencia* definovaná ako reťazec znakov z abecedy {A, C, G, T}. Tieto znaky zodpovedajú *bázam* adenín, cytozín, guanín a tymín, ktoré kódujú genetickú informáciu v bunkách organizmov.

Pri práci s DNA sekvenciami často používame pojem *inverzia*. Inverziu si môžeme predstaviť ako zobrazenie I z množiny DNA sekvencií do množiny DNA sekvencií. Inverzia jednej bázy je opäť báza, pričom $I(\mathtt{A}) = \mathtt{T}$, $I(\mathtt{C}) = \mathtt{G}$, $I(\mathtt{G}) = \mathtt{C}$ a $I(\mathtt{T}) = \mathtt{A}$. Inverziu DNA sekvencie definujeme ako $I(x_1, x_2, \ldots x_n) = I(x_n), I(x_{n-1}), \ldots, I(x_1))$ (Teda každú bázu zinvertujeme a výsledok otočíme.)

Platí, že $I(I(x)) = x$. Dve sekvencie x a y sú navzájom inverzné, ak $I(x) = y$.

Biologické pozadie za inverziou je také, že DNA sa v organizmoch vyskytuje vo forme dvojzávitnice, ktorú tvoria dve opačne orientované vlákna (teda tam, kde jedna sekvencia začína, druhá končí a naopak). V týchto sekvenciách sa vždy oproti báze A nachádza T a oproti C je G. Tým pádom pokiaľ jedno vlákno dvojzávitnice tvorí DNA sekvencia x, druhé vlákno tvorí $I(x)$.

1.2 Mutácie v DNA sekvenciách

Pri prenášaní genetickej informácie z organizmu na jeho potomkov sa DNA kopíruje. Inak povedané, potomok nejakého organizmu má len kópiu DNA sekvencie pôvodného organizmu. Pri kopírovaní DNA sekvencií môžu nastať chyby, ktoré nazývame *mutácie*.

V našej práci uvažujeme dva typy mutácií: lokálne mutácie a dlhé mutácie. Lokálne mutácie predstavujú krátke (obvykle do 10 báz) zmeny v DNA sekvencii, pričom rozlišujeme tri typy lokálnych mutácií.

- *Substitúcia* je zmena jednej bázy v DNA sekvencii.

- *Inzercia* je vloženie krátkeho úseku do DNA sekvencie.
- *Krátka delécia* je zmazanie krátkeho úseku DNA sekvencie.

Dlhé mutácie sú zmeny dlhé tisíce až desaťtisíce báz. V tejto práci budeme uvažovať nasledujúce typy dlhých mutácií.

- *Duplikácia* je skopírovanie dlhého úseku DNA sekvencie na iné miesto v sekvencii. Dôsledkom duplikácie vzniknú v sekvencii dva rovnaké úseky DNA. Pôvodný úsek nazývame originál a novovzniknutý úsek kópia. Smer duplikácie môže byť buď vľavo, pokiaľ sa kópia nachádza v sekvencii vľavo od originálu alebo, vpravo v opačnom prípade.
- *Duplikácia s inverziou* je skopírovanie dlhého úseku DNA, pričom na cieľové miesto v DNA je vložená inverzia pôvodného úseku. Podobne ako pri duplikácii rozlišujeme smer vľavo a vpravo.
- *Delécia* je zmazanie dlhého úseku DNA sekvencie.

Niekedy pojmom duplikácia označujeme spoločne duplikáciu a duplikáciu s inverziou. Všetky tri typy dlhých mutácii označujeme spoločným názvom *udalosť*.

Každá duplikácia zanechá v sekvencii dlhé rovnaké úseky DNA. Podobne duplikácia s inverziou zanechá v sekvencii dlhé invertované úseky DNA. Označme tieto úseky x a y. Tieto úseky sú kópiami nejakého úseku z, ktorý sa nachádzal v DNA sekvencii pred mutáciou. Hovoríme, že z je otec úsekov x a y. Vzťah „z je otec x“ je relácia a tranzitívnym uzáverom tejto relácie dostaneme reláciu „z je predok x“. Opačná relácia je „x je potomok y“.

1.3 Génové zhluky a ich analýza

Vo väčšine DNA sekvencie sa nedejú dlhé mutácie veľmi často. Avšak niekedy sa stane, že pre organizmy je evolučne výhodné mať viac kópií nejakého génu (gén je úsek DNA sekvencie, ktorý kóduje informáciu pre tvorbu nejakého proteínu). Aby sa tieto gény mohli rýchlejšie prispôsobovať vonkajším vplyvom. Príkladom sú gény, ktoré majú na starosti imunitu organizmov.

Takéto gény spôsobujú evolučný tlak na častejší výskyt duplikácií. To má za následok tvorbu zložitých génových zhlukov a aby sme mohli funkcii týchto génov lepšie porozumieť, potrebujeme túto komplikovanú štruktúru rozpliesť.

Našim cieľom je teda pre danú DNA sekvenciu zistiť, aké duplikácie viedli k jej vzniku.

1.4 História DNA sekvencie

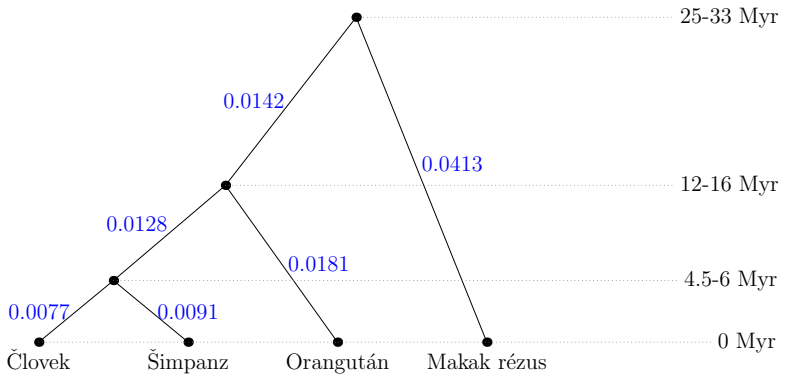
Postupnosť dlhých v DNA sekvencii budeme nazývať *duplikačná história* DNA sekvencie, skrátene *história*. Hľadanie tejto postupnosti udalostí nazývame *rekonštrukcia histórie*.

Podobu DNA sekvencie, ako vyzerá v súčasnosti nazývame *súčasná DNA sekvencia*. Podobu sekvencie pred najstaršou udalosťou nazývame *ancestrálna DNA sekvencia*.

Čas medzi najstaršou udalosťou a súčasnosťou resp. čas medzi súčasnou a ancestrálnou sekvenciou nazývame dĺžka histórie. Čím je história dlhšia, tým ťažšie je zistiť, ako história vyzerala. Totiž aby sme mohli históriu rekonštruovať, potrebujeme vedieť, ktoré úseky vznikli zo spoločného predka sériou duplikácií. To sa robí tým ťažšie, čím viac lokálnych mutácií v sekvencii nastalo, pretože sekvencie, ktoré boli predtým rovnaké sa na seba čím ďalej tým menej podobajú.

Aby sme sa nemuseli zaoberať tým, ako rýchlo sa dejú mutácie, určíme si vlastnú jednotku času. Nech *jednotka času* je taká doba, počas ktorej v DNA sekvencii dĺžky ℓ nastane v očakávanom prípade ℓ lokálnych mutácií.

V našom prípade skúmame DNA sekvenciu človeka a príbuzných primátov. Pozeráme sa približne 0.04 jednotiek času do minulosti, čo zodpovedá skutočnej dobe 25 až 33 miliónov rokov. Približne v tejto dobe žil spoločný predok človeka a makaka. Na obrázku 1.1 vidíme približné časy niektorých ďalších príbuzných organizmov [Ora11]. Dĺžky hrán na obrázku 1.1 zodpovedajú času meranom v našich jednotkách.



Obr. 1.1: Fylogenetický strom človeka a niektorých príbuzných druhov.

1.5 Atómy

Každá duplikácia rozdelí DNA sekvenciu na troch miestach, delécia na dvoch. Najdlhšie úseky DNA sekvencie, ktoré samotné ani ktorých predkovia neboli rozdelení žiadnou udalosťou voláme *atómy*.

Každému atómu priradíme typ, čo bude kladné celé číslo. Dva atómy budú mať rovnaký typ, ak vznikli zo spoločného predka a rôzny typ, ak nevznikli zo spoločného predka. Takto sa nám rozpadne množina atómov do tried ekvivalencie podľa typu.

Keď rozdelíme DNA sekvenciu na atómy, dostaneme takzvanú *sekvenciu atómov*. Zo súčasnej DNA sekvencie dostaneme *súčasnú sekvenciu atómov*. Nasekaním ancestrálnej DNA sekvencie dostaneme *ancestrálnu sekvenciu atómov*. V ancestrálnej sekvencii atómov sa nachádza atóm každého typu najviac raz.

1.6 Duplikačná udalosť

Zatiaľ čo duplikácie po sebe zanechávajú stopy v podobe podobných úsekov, delécie tieto stopy mažú, čím sťažujú rekonštrukciu histórie.

Väčšinu delécii nedokážeme zrekonštruovať, pretože keď sa v súčasnej sekvencii po nich nenachádzajú žiadne stopy, nemáme ako zistiť, že vôbec nastali. Deléciu vieme odhaliť len vtedy, ak nastala vo vnútri nejakého úseku skopírovaného duplikáciou. Len ak nájdeme dva dlhé podobné úseky a v jednom nejaká časť (príp. nejaké) časti chýbajú, môžeme sa domnievať, že išlo o dôsledok delécie. Každú deléciu, ktorú vieme odhaliť môžeme takto priradiť k nejakej duplikácii.

Postupnosť jednej duplikácie (s inverziou alebo bez inverzie) a niekoľkých delécií, ktoré odstránili podúseky duplikácie nazývame *duplikačná udalosť*.

Kapitola 2

Algoritmus na rekonštrukciu histórií

2.1 Problém rekonštrukcie histórie

Definícia 2.1 (Problém rekonštrukcie histórie)**.** Problém rekonštrukcie histórie je nájsť pre súčasnú DNA sekvenciu S, ktorá vznikla z ancestrálnej DNA sekvencie S_0 postupnosťou lokálnych mutácií M a udalostí U, jej históriu H. Riešenie problému je správne, pokiaľ množina prvkov postupnosti udalostí U je rovnaká ako množina udalostí H.

Teoreticky teda nezáleží na poradí udalostí v H, no nie každé poradie dáva zmysel. O poradí udalostí a vôbec o tom, kedy dve udalosti považujeme za rovnaké si povieme viac v podkapitole 2.5.

V našej práci sa zaoberáme riešením problému rekonštrukcie v prípade, že poznáme len súčasnú sekvenciu S. Problém riešime pravdepodobnostne, pričom sa snažíme, aby pravdepodobnosť, že algoritmus nájde správne riešenie bola čo najväčšia vzhľadom na náhodné S_0, M a U, ktorých náhodné distribúcie sú dané pravdepodobnostným modelom aproximujúcim skutočnú evolúciu DNA sekvencií (viď podkapitolu 4.5).

Riešenie tohto problému delíme na dve časti.

Prvá časť je navrhnúť algoritmus, ktorý vygeneruje množinu histórií danej sekvencie, bez priradených časov k udalostiam tak, aby sa s veľkou pravdepodobnosťou v tejto postupnosti nachádzala aj správna história, hoci nemusíme povedať, ktorá to je.

Druhá časť problému je pre danú DNA sekvenciu a množinu histórií určiť, ktorá z histórií je tá skutočne správna.

Primárne sa v tejto práci budeme zaoberať prvou časťou problému rekonštrukcie. Navrhneme, implementujeme a dôkladne preskúmame algoritmus, ktorý tento problém bude riešiť. Budeme sa snažiť, aby algoritmus dokázal čo najrýchlejšie, resp. s čo najmenším počtom pokusov nájsť správnu históriu.

To, ako sa algoritmu darí, budeme vedieť vyhodnotiť zatiaľ len na simulovaných dátach, keďže len pre tie vieme presne povedať, aká história je správna a teda či ju algoritmus našiel.

2.2 Rozdelenie sekvencie na atómy

Namiesto toho, aby sme pracovali so samotnou DNA sekvenciou, budeme rekonštruovať históriu sekvencie atómov. Sekvencia atómov je totiž výrazne kratšia (20 až 200 atómov v porovnaní s 200 000 až 400 000 bázami DNA sekvencie), čo vedie k podstatne lepšej časovej zložitosti algoritmu. Navyše, keď máme sekvenciu atómov, umožňuje nám to abstrahovať od lokálnych mutácií a sústrediť sa len na udalosti, prípadne naopak, pozrieť sa na lokálne mutácie bez toho, aby sme sa museli zaoberať dlhými udalosťami.

Udalosti totiž vždy kopírujú/mažú zo sekvencie len celé kusy atómov, a dôsledky lokálnych mutácií vidíme len medzi atómami rovnakých typov.

Problémom segmentácie sekvencie na atómy sa zaoberali Brejová et. al. [BBV11] a Višňovská et. al. [VVB13]. Výsledkom segmentácie je zoznam atómov, v poradí v akom sa za sebou nachádzajú v sekvencii, pričom ku každému atómu máme priradený *orientovaný typ*.

Definícia 2.2 (Orientovaný typ)**.** Orientovaný typ atómu je kladné alebo záporné číslo, pričom atómy, ktoré majú spoločného predka majú absolútnu hodnotu typu rovnakú. Atómy, ktorých DNA sekvencie sú navzájom inverzné majú opačné znamienko a atómy, ktorých DNA sekvencie nie sú navzájom inverzné majú rovnaké znamienko.

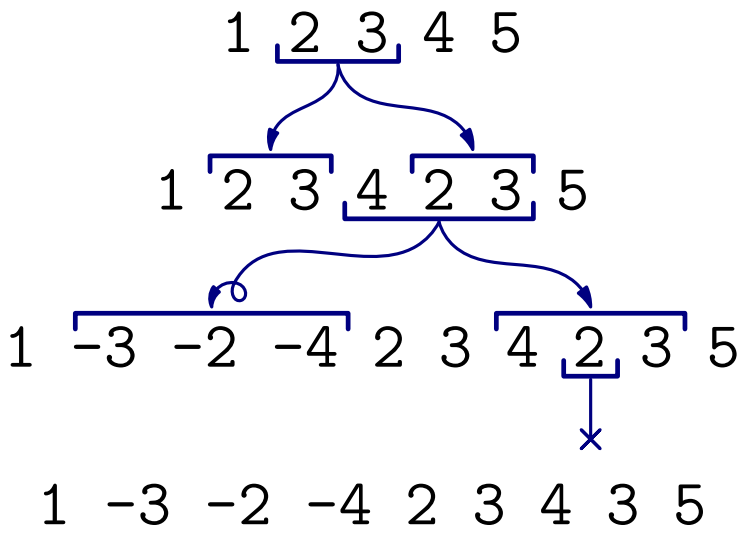
Napriek tomu, že segmentáciu na atómy vieme robiť pomerne presne, pre krátke atómy nedokážeme spoľahlivo povedať, či vznikli zo spoločného predka alebo nie. Nedokážeme pre ne s dostatočnou istotou povedať, či prípadná podobnosť ich DNA sekvencií nie je len dôsledok náhody. Z toho dôvodu odstránime zo sekvencie všetky atómy, ktorých dĺžka je kratšia ako 50 báz.

Na obrázku 2.1 môžeme vidieť príklad sekvencie atómov aj s jej históriou

2.3 Schéma algoritmu

Algoritmus postupne vygeneruje zvolený počet histórií, pričom každú históriu rekonštruuje samostatne, nezávisle od ostatných.

Rekonštrukciu jednej histórie robíme odzadu (t.j. od súčasnej sekvencie) po jednotlivých duplikačných udalostiach – vždy sa pre aktuálnu sekvenciu atómov snažíme uhádnuť, aká udalosť nastala posledná, následne vypočítame, ako vyzerala sekvencia



Obr. 2.1: Na obrázku môžeme vidieť príklad histórie sekvencie atómov. V tejto histórii nastala postupne duplikácia, duplikácia s inverziou a delécia.

atómov pred danou udalosťou. Toto opakujeme, až kým sa v sekvencii atómov nevyskytuje každý druh atómu práve raz.

Táto základná kostra nie je ničím novým, rovnakým spôsobom sme rekonštruovali histórie už v predošlej práci.

Najdôležitejšia časť algoritmu, na ktorej závisí jeho celková úspešnosť, je spôsob, akým sa pre danú sekvenciu atómov určí, ktorá udalosť pravdepodobne nastala ako posledná. Práve toto je hlavné miesto, ktorému venujeme najviac úsilia, pretože algoritmu sa podarí úspešne zrekonštruovať celú históriu len ak správne uhádne udalosť v každom kroku. Ak sa prvýkrát pomýli, už nemôže nič spraviť, aby históriu zrekonštruoval správne. Preto sa v našej práci usilujeme o čo najvyššiu pravdepodobnosť uhádnutia správnej udalosti. Treba poznamenať, že správna udalosť nemusí byť vždy len jedna. Pokiaľ dve udalosti na sebe navzájom nezávisia a ovplyvňujú disjunktné časti sekvencie, je pre nás jedno, v akom poradí tieto udalosti vykonáme. Tým pádom môžeme obe považovať za správne voľby pri rekonštrukcii histórie.

2.4 Výber poslednej udalosti

Ako teda algoritmus vyberá, aká udalosť nastala ako posledná?

Základná myšlienka je priradiť každej udalosti u nejaké skóre $s(u)$, ktoré vyjadruje, ako veľmi si myslíme, že udalosť u je tá správna. Následne zvolíme náhodnú udalosť tak, že pravdepodobnosť $P(u)$ výberu udalosti u je daná vzťahom 2.1, kde U je množina všetkých udalostí.

$$P(u) = \frac{s(u)}{\sum_{u' \in U} s(u')} \tag{2.1}$$

Keďže udalostí, ktoré mohli teoreticky nastať je príliš veľa, prvé čo program spraví je, že vyberie spomedzi všetkých možných udalostí malú množinu kandidátov, s ktorými bude naďalej pracovať. Týchto kandidátov vyberieme na základe jednoduchšej skórovacej funkcie, algoritmom popísaným v kapitole 3.

Tento algoritmus však nevzorkuje množinu, ale postupnosť kandidátov, takže, aby sme skutočne dostali množinu, vyhodíme z postupnosti duplikáty. Smer duplikácií (viď podkapitolu 1.2), ktoré algoritmus vzorkuje je náhodný, takže pridáme nových kandidátov, aby každá udalosť, ktorá bola v pôvodnej množine v aspoň jednom smere sa nachádzala v novej množine v oboch smeroch. S takto zostojenou množinou kandidátov prejdeme do druhej fázy.

V druhej fáze pre každého kandidáta v množine kandidátov vypočítame zložitejšou a presnejšou skórovacou funkciou nové skóre, podľa ktorého pravdepodobnostne vyberieme výslednú udalosť. Toto druhé skórovanie si vysvetlíme v kapitole 4.

2.5 Porovnávanie udalostí a histórií

Aby sme mohli hodnotiť, či je náš algoritmus správny, potrebujeme definovať, kedy považujeme históriu za správnu, kedy považujeme udalosť za správnu a podobne.

Kľúčové je definovať kedy sú dve udalosti rovnaké a zvyšok sa bude odvíjať od tejto definície.

Základný princíp, ktorého sa chceme držať je, že veci, ktoré sa dajú rozlíšiť bez znalosti správnej histórie, by sme mali považovať za rôzne. Napríklad atómy majú vo všeobecnosti rôzne DNA sekvencie, a to aj v rámci jedného typu, takže medzi nimi budeme rozlišovať.

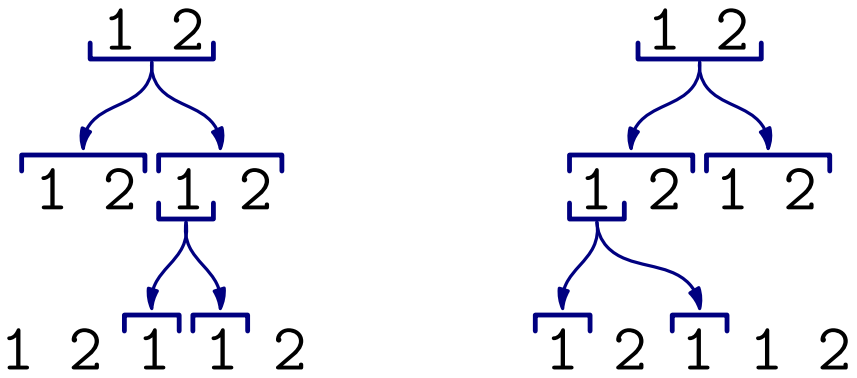
Každému atómu v súčasnej sekvencii priradíme jednoznačný identifikátor. Ostatné atómy (predkovia súčasných atómov) budú identifikované množinou identifikátorov atómov zo súčasnej sekvencie, ktoré z neho vzniknú postupnosťou duplikácií. Napríklad priamy predok atómov A, B bude $\{A, B\}$, pričom budeme používať skrátený zápis AB. Priamy predok atómu AB a atómu C by bol ABC.

Definícia 2.3. Dva atómy sú rovnaké, pokiaľ majú rovnaký typ aj rovnaký identifikátor. Porovnávanie typu potrebujeme zaviesť kvôli deléciám, pretože zmazaný atóm nemá potomkov a jeho identifikátor je prázdna množina.

Definícia 2.4. Dve delécie považujeme za **rovnaké**, pokiaľ zmazali rovnakú množinu atómov. Dve duplikácie bez inverzie považujeme za **rovnaké**, pokiaľ oboma dupli-

káciami vznikli rovnaké množiny atómov. Dve duplikácie s inverziou považujeme za **rovnaké**, pokiaľ oboma duplikáciami vznikli rovnaké množiny atómov. Udalosti rôznych typov nie sú rovnaké, čiže žiadna delécia nie je rovnaká so žiadnou duplikáciou a žiadna duplikácia bez inverzie nie je rovnaká s duplikáciou s inverziou.

Všimnime si, že byť rovnaký je relácia ekvivalencie. Príklady rôznych udalostí môžeme vidieť na obrázku 2.2. Pokiaľ si označíme atómy v spodnej sekvencie $1_A, 2_B, 1_C, 1_D, 2_E$, môžeme ľahko uvidieť rozdiel v spodných duplikáciách. Duplikácia vľavo dole vytvorí atómy $1_C, 1_D$ a duplikácia vpravo dole vytvorí atómy $1_A, 1_C$. Avšak ani vrchné duplikácie nie sú rovnaké. Množiny atómov, ktoré vytvoria sú $\{1_A, 2_B, 1_{CD}, 2_E\}$ a $\{1_{AC}, 2_B, 1_D, 2_E\}$. Tieto histórie by sme od seba vedeli odlíšiť, keby sme sa pozreli na DNA sekvencie atómov, jedna z nich by bola vierohodnejšia (viď podkapitolu 3.2.



Obr. 2.2: Tieto zdanlivo podobné histórie sú v skutočnosti rôzne. Líšia sa dokonca v oboch udalostiach.

Na obrázku 2.3 vidíme príklad histórií, ktoré nevieme odlíšiť, pre obe môžu súčasné sekvencie vyzerať identicky. Na základe definície 2.4 sú udalosti v týchto históriách rovnaké. Podobne, keby sme mali viac nezávislých udalostí, napríklad dve duplikácie `1 2 3 4 5 6` $\rightarrow$ `1 2 3 2 4 5 6` $\rightarrow$ `5 1 2 3 2 4 5 6`, nevieme povedať ani poradie, v akom udalosti nastali.

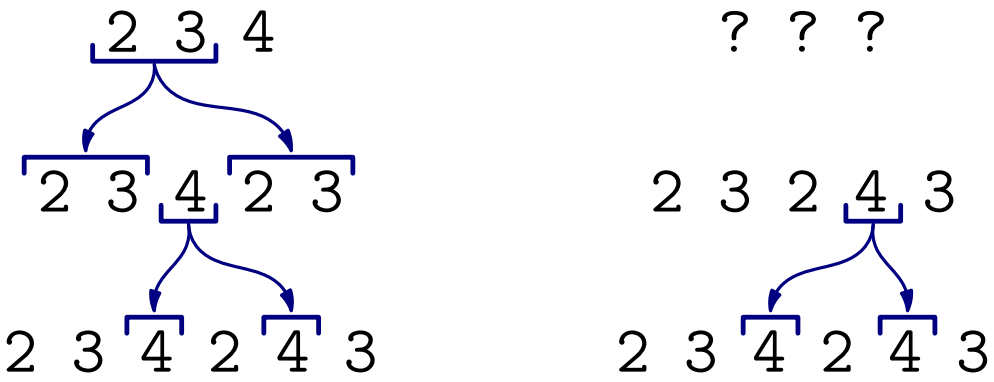
Obr. 2.3: Takmer každá história s jedinou duplikáciou vyzerá ako história vľavo alebo história vpravo. Napriek tomu nikdy nedokážeme povedať, ktorá z nich skutočne nastala. Budeme teda považovať obe za správne.

Definícia 2.5. Dve histórie sú rovnaké, ak množiny ich udalostí sú rovnaké.

Ak H je skutočná história nejakej sekvencie, tak o H' budeme hovoriť, že je správna, pokiaľ je rovnaká ako H. O udalosti u hovoríme že je správna, ak je rovnaká ako nejaká udalosť v H.

Sú však prípady, kedy nám nestačí určiť správnu udalosť. Napríklad, ak začneme rekonštruovať sekvenciu `2 3 4 2 4 3` ktorá vznikla históriou na obrázku 2.4, nestačí nám uhádnuť správnu udalosť ale potrebujeme uhádnuť aj či jej smer bol vľavo alebo vpravo.

Definícia 2.6. Dve udalosti sú striktne rovnaké, ak sú rovnaké a majú rovnaký smer. Udalosť je striktne správna, ak je striktne rovnaká s nejakou udalosťou v správnej histórii.



Obr. 2.4: Príklad histórie, kedy záleží na správnom smere udalosti pri rekonštrukcii. Vľavo je skutočná história, vpravo nevydarený pokus o rekonštrukciu.

Kapitola 3

Generovanie kandidátov na udalosti

3.1 Algoritmus na rýchle vzorkovanie kandidátov

Kravec vo svojej práci [Kra11] poukázal na to, že duplikačných udalostí, ktoré mohli nastať, je príliš veľa na to, aby sa s nimi dalo v praxi dobre pracovať. Nech n je počet atómov v súčasnej sekvencii. Ak by sme neuvažovali delécie, je možných udalostí $\Theta(n^3)$, s maximálne jednou deléciou dĺžky 1 je ich $\Theta(n^4)$, s maximálne jednou ľubovoľne dlhou deléciou je $\Theta(n^5)$ duplikačných udalostí ...

Kravec preto navrhol jednoduchú skórovaciu funkciu, ktorá každej duplikačnej udalosti u priradí skóre

$$s(u) = p_{dup}^{\ell} \cdot p_{del}^{k} \cdot q_{del}^{d_1+\cdots+d_k} \tag{3.1}$$

kde ℓ je dĺžka duplikácie (počet duplikovaných atómov), k je počet delécií, ktoré nasledovali po duplikácií a $d_1, \ldots, d_k$ sú dĺžky týchto delécií. p_{dup}, p_{del} a q_{del} sú parametre algoritmu, v našom prípade $p_{dup} = 2$, $p_{del} = 0.05$ a $q_{del} = 0.5$.

Zároveň Kravec ukázal rýchly algoritmus, ktorý dokáže pravdepodobnostne vzorkovať duplikačné udalosti, pričom pravdepodobnosť výberu udalosti u spomedzi množiny všetkých možných udalostí U je

$$P[u] = \frac{s(u)}{\sum\limits_{u' \in U} s(u')} \tag{3.2}$$

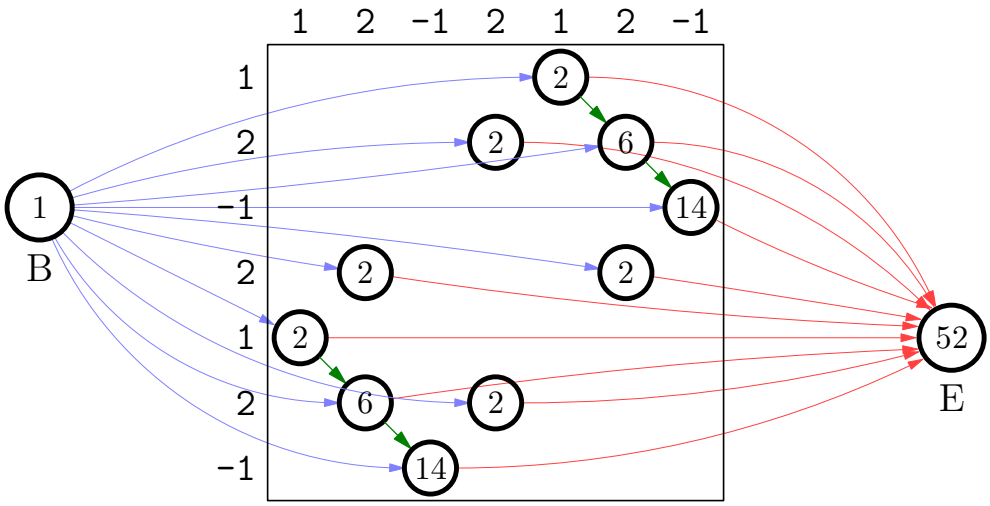
Tento algoritmus dokáže efektívne vzorkovať udalosti bez potreby explicitne vyskúšať a jednotlivo vyhodnotiť všetkých kandidátov z U.

Ukážeme si najprv, ako by vyzeral algoritmus, keby sme uvažovali len duplikácie bez inverzie a skórovacia funkcia by bola $s(u) = 2^{\ell}$. Napríklad v sekvencii atómov `1 2 -1 2 1 2 -1`, by udalosť, ktorá duplikovala atómy `1 2 -1` mala skóre 8, zatiaľ čo udalosti, ktoré duplikujú len atómy `1 2` alebo `2 -1` majú skóre len 4. Udalosti, ktoré by

skopírovali len jeden atóm by mali skóre 2 a boli by vzorkovacím algoritmom vybraté so štyrikrát menšou pravdepodobnosťou ako najdlhšia udalosť.

Zostrojíme si orientovaný ohodnotený graf, ktorého vrcholy budú dvojice (i, j) také, že $i \neq j$ a zároveň atóm na pozícii i má rovnaký orientovaný typ (t.j. vrátane znamienka) ako atóm na pozícii j. Okrem toho budú v grafe ešte dva vrcholy, ktoré označíme B a E. Z vrchola B vedie hrana do každej dvojice (i, j), ktorá je vrcholom grafu, a tiež z každej takejto dvojice vedie hrana do vrchola E. Hrany z (i, j) do E majú cenu 1 a hrany z B do (i, j) majú cenu 2. Ak obe dvojice (i, j) aj $(i + 1, j + 1)$ sú vrcholmi grafu, tak medzi nimi vedie hrana s cenou tiež 2. Cenu cesty v tomto grafe definujeme ako súčin cien hrán na ceste.

Príklad takéhoto grafu pre sekvenciu atómov `1 2 -1 2 1 2 -1` môžeme vidieť na obrázku 3.1. Hodnoty vo vrcholoch sú súčty cien všetkých ciest, ktoré vedú z B do týchto vrcholov.



Obr. 3.1: Príklad grafu pre sekvenciu atómov `1 2 -1 2 1 2 -1`. Hrany vedúce do vrchola E (červené) majú cenu 1. Ostatné hrany (zelené a modré) majú cenu 2.

Dôvod, prečo tento graf používame je, že každá cesta z vrchola B do vrchola E zodpovedá jednej duplikácii bez inverzie (a obrátene, každej duplikácii zodpovedá jedna cesta). Navyše cena takejto cesty je rovná 2^ℓ, kde ℓ je dĺžka duplikácie. Čiže vybrať náhodnú duplikáciu s pravdepodobnosťou úmernou 2^ℓ znamená vybrať náhodnú cestu v tomto grafe s pravdepodobnosťou úmernou jej cene. To vieme robiť nasledovne.

Pre každý vrchol v spočítame súčet cien všetkých ciest vedúcich z B do v. Potom náhodnú cestu skonštruujeme tak, že začneme vo vrchole E a hýbeme sa proti smeru hrán. Hranu, ktorou sa vydáme vždy vyberieme náhodne, s pravdepodobnosťami určenými hodnotami vo vrcholoch, do ktorých sa môžme pohnúť.

Algoritmus sa dá ľahko rozšíriť tak, aby vzorkoval aj duplikácie s inverziou a duplikačné udalosti s deléciami. V Kravcovej práci [Kra11] je toto rozšírenie detailne vysvetlené, takže tu uvádzame len hlavnú ideu.

Nový graf bude mať o rozmer viac, bude obsahovať vrcholy B a E a okrem nich bude v grafe 6 poschodí po $n \times n$ vrcholoch. Vrcholy na poschodí p označujeme (p, i, j). Každý vrchol reprezentuje nejakú časť duplikačnej udalosti, pričom cesta v grafe zodpovedá duplikačnej udalosti, ktorá vznikne spojením týchto častí.

Napríklad význam vrchola $(1, i, j)$ je taký, že atómy na pozíciách i a j v sekvencii atómov vznikli spoločnou duplikáciou. Do týchto vrcholov vedú hrany len vtedy, ak tieto atómy majú rovnaký typ a tieto hrany vedú len z vrcholov B, $(1, i-1, j-1)$, $(2, i-1, j-1)$ a $(3, i-1, j-1)$.

Vrchol $(2, i, j)$ zodpovedá delécii atómu v druhej kópii duplikovanej sekvencie. Tento atóm sa nachádza len v prvej kópii, čiže do $(2, i, j)$ vedie hrana len z $(1, i-1, j)$, $(2, i-1, j)$ a $(3, i-1, j)$. Obrazne hovoríme, že sa pomocou týchto hrán pohybujeme len v prvej kópii duplikovanej sekvencie. Hrany medzi poschodiami môžeme interpretovať ako začiatok delécie alebo koniec delécie.

Uveďme si príklad. Majme sekvenciu atómov `1 2 3 4 5 1 1 2 5`, a zaujíma nás cesta v grafe, ktorá zodpovedá duplikačnej udalosti, ktorá najprv duplikuje `1 2 3 4 5` a následne v druhej kópii nastane delécia `3 4`. Cesta vedie postupne cez vrcholy B, $(1, 1, 7)$, $(1, 2, 8)$, $(2, 3, 8)$, $(2, 4, 8)$, $(1, 5, 9)$, E.

Analogicky, poschodie 3 bude zodpovedať delécii v prvej kópii. Poschodia 4, 5 a 6 budú slúžiť pre duplikáciu s inverziou a delécie v nej. Hrany na týchto poschodiach budú mať opačný smer v jednom rozmere, napríklad môže viesť hrana z $(p, i-1, j+1)$ do $(4, i, j)$ pre $p \in 4, 5, 6$, čo je však možné len v prípade, že atómy na pozíciách i a j mohli vniknúť duplikáciou s inverziou, t.j. majú opačný orientovaný typ.

Ceny hrán vedúce do vrcholov na 1. a 4. poschodí budú p_{dup}. Ceny hrán v rámci poschodí 2, 3, 5, 6 budú q_{del} a hrany vedúce do týchto poschodí z poschodí 1 a 4 budú mať cenu $p_{del} \cdot q_{del}$.

Časová zložitosť navzorkovania k kandidátov zo sekvencie, ktorá má n atómov je $O(n^2 + nk)$. Aby celková časová zložitosť vzorkovania bola $O(n^2)$, budeme generovať budeme generovať $O(n)$ kandidátov. V podkapitole 5.3 experimentálne odhadneme presnejší počet kandidátov potrebný na to, aby sa medzi kandidátmi pravdepodobne nachádzala aj správna udalosť.

Nevýhodou algoritmu je, že vzorkuje kandidátov len podľa dĺžky duplikácie, počtu delécií a dĺžok delécií. Preto neskôr v tejto kapitole upravíme ceny hrán v spomínanom grafe s využitím dodatočných informácií o atómoch, čo nám umožní podobným algoritmom vzorkovať udalosti na základe lepšej skórovacej funkcie.

3.2 Využitie DNA sekvencií atómov pri rekonštrukciu histórií

Keď zoberieme dve rovnaké sekvencie a necháme ich nezávisle sa vyvíjať, teda necháme, aby sa v sekvenciách nezávisle diali lokálne mutácie, tak sa tieto sekvencie postupom času začnú odlišovať. Čím dlhšie sa tieto sekvencie vyvíjajú oddelene, tým väčší je očakávaný počet pozícií, na ktorých sa tieto sekvencie líšia. Presnejšie, podľa Jukes-Cantorovho modelu opisovaného v podkapitole 4.5, môžeme vypočítať očakávaný počet rozdielov v dvoch pôvodne rovnakých sekvenciách dlhých ℓ po čase t ako $\frac{3\ell}{4}\left(1 - e^{-4t/3}\right)$.

Táto úvaha sa dá použiť aj opačným smerom. Na základe toho, ako veľmi sú sekvencie podobné, dokážeme odhadnúť pred akým časom mohli mať tieto sekvencie spoločného predka. Tento odhad je tým presnejší, čím dlhšie sú dané sekvencie. Bohužiaľ, pri krátkych sekvenciách a tiež veľmi krátkom čase vývoja je relatívna disperzia počtu lokálnych mutácií príliš veľká na spoľahlivý odhad času.

Avšak aj takáto nespoľahlivá informácia môže pomôcť pri rekonštrukcii histórie a to, ako veľmi pomôže, sme sa snažili preskúmať v predošlej práci [Hoz14]. Otestovali sme, ako sa zmení úspešnosť rekonštrukcie histórie, keď pri výbere udalosti spomedzi kandidátov budeme zvýhodňovať duplikácie, ktorých duplikované atómy mali podobnejšie DNA sekvencie. Výsledkom bolo, že táto metóda mala pozitívny vplyv na nájdenie histórie s veľkou vierohodnosťou, nie však na nájdenie správnej histórie. V tejto práci sa však sústredíme práve na správnosť histórie, pretože sme zistili, že vierohodnosť nie je dobrá metrika (viď. podkapitola **??**).

Navyše pri počítaní podobnosti sa vyskytlo niekoľko problémov.

Prvý problém pri práci s DNA sekvenciami atómov je ten, že síce poznáme, ako tieto sekvencie vyzerajú v súčasnosti, ale nevieme povedať, ako vyzerala sekvencia atómu pred nejakým časom, prípadne ako vyzerala sekvencia spoločného predka dvoch atómov. Predtým sme sa s tým snažili vysporiadať tak, že sme pre každú pozíciu sekvencie mali štyri reálne čísla, ktoré postupne určovali, ako veľmi si myslíme, že je na danej pozícii písmeno `A`, písmeno `C`, písmeno `G` alebo písmeno `T`. Dôsledok však bol, že akonáhle sme sa príliš vzdialili od súčasnosti, už sme v podstate nedokázali spoľahlivo povedať nič.

Druhý problém bol s tým, že vyhodnocovať podobnosť dvoch atómov a aktualizovať informácie po spracovaní udalosti bolo príliš časovo náročné. Kým časová zložitosť iných častí algoritmu závisela len od počtu atómov v sekvencii (20 až 200 atómov), zložitosť tejto metódy závisela od dĺžky celej DNA sekvencie (200 000 až 400 000 báz). Aby sme rekonštruovali histórie dostatočne rýchlo, museli sme od tejto metódy upustiť.

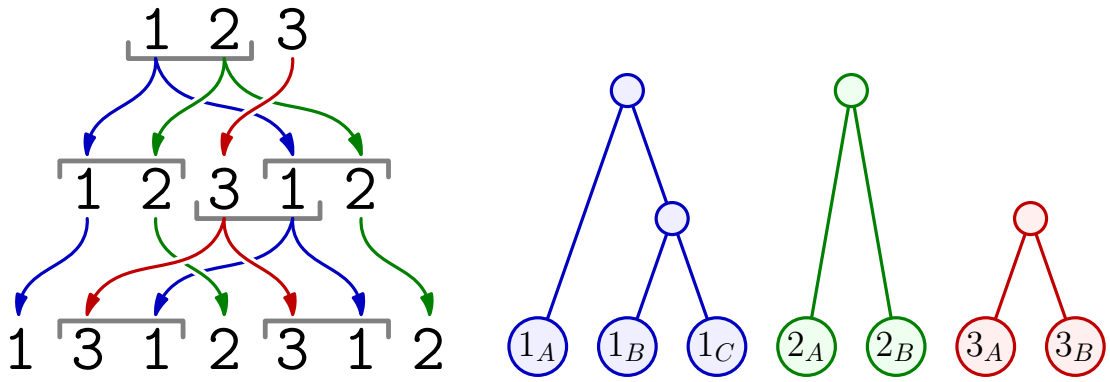
Tretí problém bola príliš veľká nepresnosť odhadovania času spoločného predka, po-

kiaľ sa sekvencie vyvíjali krátko a nastalo len malé množstvo lokálnych mutácií. Najmä v takýchto prípadoch, ale aj celkovo pri rekonštruovaní histórie nie je dôležité len na *koľkých* pozíciách sa DNA sekvencie odlišujú ale aj na *ktorých* pozíciách. Toto je znázornené na obrázkoch 3.3 a 3.4 v podkapitole 3.3. Pôvodná metóda, ktorá zohľadňovala len počty rozdielnych pozícií v sekvenciách, mala z týchto dôvodov veľkú chybovosť.

Týmto problémom sme sa vyhli použitím odlišnej metódy, ktorá oveľa lepšie zachycuje vzájomné vzťahy medzi atómami.

3.3 Evolučné stromy atómov

Ak poznáme históriu DNA sekvencie a poznáme tiež rozdelenie DNA sekvencie na atómy, vieme pre každý typ atómu určiť jeho takzvaný *zakorenený evolučný strom*. Tento strom zachytáva udalosti, ktoré sa týkali atómov jedného typu.



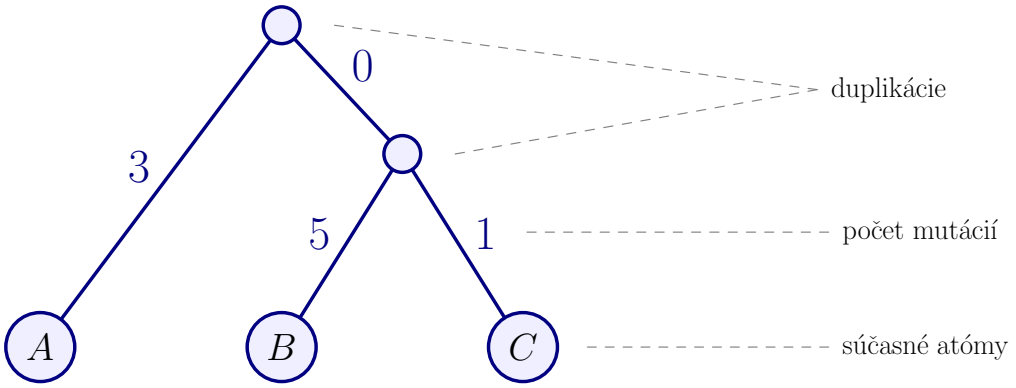
Obr. 3.2: Vľavo je príklad histórie s dvoma duplikáciami. V prvom riadku je ancestrálna sekvencia, v treťom súčasná sekvencia atómov. Vpravo od histórie sú farebne odlíšené evolučné stromy atómov, postupne pre typ 1, typ 2 a typ 3.

Listy evolučného stromu zodpovedajú atómom vyskytujúcim sa v súčasnej sekvencii. Každý vnútorný vrchol v zodpovedá atómu tesne pred nejakou duplikáciou u. Takýto vrchol má dvoch synov, ktorí zodpovedajú atómom vytvoreným duplikáciou u z atómu v. Koreň stromu zodpovedá najstaršej duplikácii atómov daného typu. V evolučnom strome sa nachádzajú len duplikačné udalosti, ktoré atóm naozaj duplikovali. Pokiaľ sa najprv atóm skopíroval a potom sa jedna z kópií zmazala, v strome tieto udalosti nebudú.

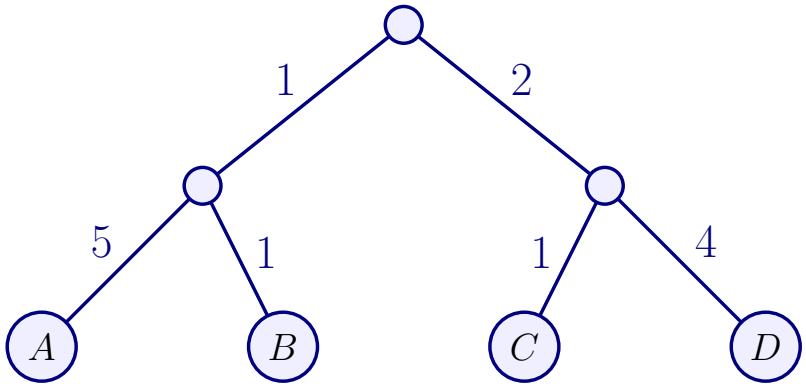
Príklad jednoduchej histórie a jej zodpovedajúcim evolučným stromom atómov môžeme vidieť na obrázku 3.2.

Hrany medzi vrcholom x a jeho otcom y, môžeme ohodnotiť počtom lokálnych mutácií, ktoré nastali v atóme x. Príklad stromu s ohodnotenými hranami je na obrázku

3.3.



Obr. 3.3: Na obrázku je príklad evolučného stromu pre atómy A, B a C. Na tomto príklade môžeme vidieť, prečo sa nemôžeme spoľahnúť len na počet rozdielov v DNA sekvenciách. Pre ilustráciu predpokladajme, že žiadne dve mutácie nenastali na rovnakej pozícii. Potom sekvencie atómov A a C sa líšia na štyroch pozíciách. Príbuznejšie atómy B a C majú až šesť rozdielov.



Obr. 3.4: Podobne ako na obrázku 3.3, ak by sme posudzovali len podľa *počtu* rozdielov v sekvenciách, atómy B, C (5 rozdielov) by sa zdali príbuznejšie ako atómy A, B (6 rozdielov). Ak by sme však posudzovali podľa *pozícií* rozdielov, zistíme, že (A, B) sa odlišujú od (C, D) na troch *spoločných* pozíciách, zatiaľ čo medzi dvojicami (A, D) a (B, C) pravdepodobne nie sú žiadne spoločné rozdiely.

Každá lokálna mutácia nastala na hrane medzi nejakými dvoma vrcholmi, čo spôsobuje, že atómy, ktoré sú v evolučnom strome blízko, zvyknú byť vzájomne podobnejšie. Ako však môžeme vidieť na obrázku 3.4, je rozdiel vyjadriť príbuznosť atómov evolučným stromom a číselným vyjadrením podobnosti.

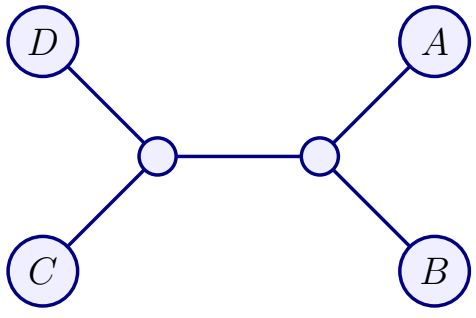
3.4 Využitie evolučných stromov atómov

Ak by sme pre každý typ atómu poznali evolučný strom atómov daného typu, bola by rekonštrukcia histórie celej sekvencie atómov oveľa jednoduchšia. Týmto sa zaoberali Vinař et. al [VBSS10]. V práci skúšali rekonštruovať históriu tak, že si najprv pre každý typ atómu zvolili, ktorý z jeho možných stromov je ten správny, a následne zrekonštruovali históriu konzistentnú so stromami všetkých typov atómov. Celý proces opakovali s rôznymi voľbami evolučných stromov. Na správne zrekonštruovanie histórie však potrebovali správne uhádnuť každý strom, takže pravdepodobnosť, že našli skutočne správnu históriu bola veľmi nízka.

V tejto práci budeme stromy atómov tiež využívať, ale namiesto toho, aby sme sa fixovali na konkrétnu podmnožinu stromov, budeme si pri rekonštrukcii histórie pomáhať štatistikami získanými zo stromov pre jednotlivé typy atómov.

Stromy, s ktorými budeme pracovať, sú *nezakorenené evolučné stromy atómov*, pretože bez znalosti histórie nie je možné koreň stromu určiť. Opäť, ak by sme predpokladali, že za rovnakú dobu nastane rovnaký počet lokálnych mutácii, našli by sme koreň ako vrchol, ktorý má na ceste ku všetkým listom približne rovnako veľa týchto mutácií. No pri krátkych časoch a krátkych DNA sekvenciách, aké sa v našich dátach môžu vyskytnúť, by tento postup ľahko viedol k nesprávne zvolenému koreňu. Ešte väčšie problémy by následne mal algoritmus na skutočných biologických dátach, pretože tam nie je až také výnimočné, že niektoré kópie génov sa vyvíjajú inak rýchlo ako iné, a teda v jednej vetve by nastalo viac lokálnych mutácií ako v inej.

Výhodou nezakorenených evolučných stromov je, že abstrahujú od toho, ako rýchlo sa dejú mutácie v ktorej časti DNA a napriek tomu dobre zachytávajú príbuznosť sekvencií. Jednoduchý príklad nezakoreného evolučného stromu môžeme vidieť na obrázku 3.5.



Obr. 3.5: Nezakorenený evolučný strom atómov z obrázku 3.4

Dobrou správou tiež je, že evolučné stromy atómov dokážeme modelovať aj vtedy, keď nepoznáme históriu sekvencie. Stačí nám poznať DNA sekvencie súčasných atómov. Tomuto problému sa venovali Huelsenback et. al [HR03], ktorí vytvorili program

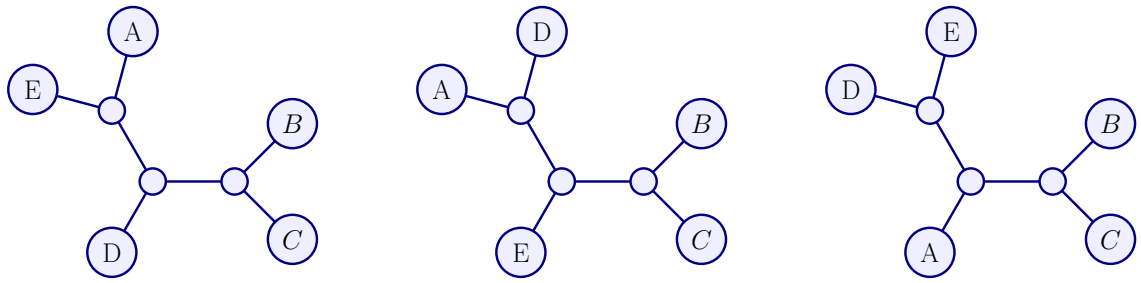
MrBayes.

Vďaka tomuto programu dokážeme pre dané DNA sekvencie atómov zistiť nielen zoznam možných nezakorenených evolučných stromov pre tieto atómy, ale aj odhadovanú aposteriórnu pravdepodobnosť každého z týchto stromov. (Teda keby sme pravdepodobnostne simulovali evolúciu, generovali takto všetky možné evolučné stromy a následne by sme vybrali len tie evolúcie, ktoré by viedli k zadaným DNA sekvenciám, aká je pravdepodobnosť, že vygenerovaný evolučný strom bol tento?)

Aposteriórne pravdepodobnosti určujeme pre každý typ atómov osobitne na základe 10 000 iterácií Metropolis-Hastingsového Markov Chain Monte Carlo (MCMC) algoritmu [HAS70], pričom prvých 2 500 iterácii bolo zahodených a následne bol vybratý každý desiaty navzorkovaný strom. MCMC algoritmus je pravdepodobnostná metóda, takže nám poskytuje len odhad aposteriórnych pravdepodobností.

Príklad stromov, ktoré môžeme pomocou programu MrBayes dostať, sú tri stromy na obrázku 3.6, ku ktorým by postupne mohli byť priradené pravdepodobnosti napríklad 0.863, 0.069 a 0.068. Na týchto stromoch môžeme pozorovať niekoľko veci. Napríklad, v žiadnom strome nemajú atómy A a B spoločného suseda. Tým pádom odhadovaná pravdepodobnosť, že tieto atómy vznikli spoločnou duplikáciou, je nula. Podobne je to aj pre ostatné kombinácie atómov A, D, E s atómami B, C. Takáto informácia nám výrazne zníži počet duplikácií, ktoré má zmysel skúšať.

Duplikácia, ktorá by vytvorila atómy D a E alebo D a A, tiež nevyzerá pravdepodobne. Naopak udalosť, ktorá by duplikovala neznámy atóm na A, E alebo B, C prípadne taká, ktorá vytvorí obe dvojice by mohla byť dobrým kandidátom na poslednú udalosť. Celková odhadovaná pravdepodobnosť, že atómy B a C majú v skutočnom evolučnom strome spoločného suseda, je 1. To však ešte nemusí znamenať, že tieto atómy vznikli spoločnou duplikáciou, pretože teoreticky na hrane medzi B a jeho susedom, alebo C a jeho susedom mohol byť pôvodne koreň stromu.



$p_1 = 0.863$ $p_2 = 0.069$ $p_3 = 0.068$

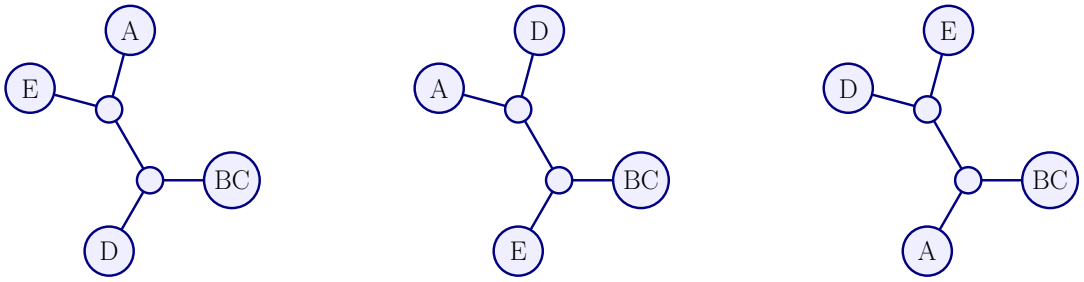
Obr. 3.6: Príklad nezakorenených evolučných stromov s priradenými aposteriórnymi pravdepodobnosťami.

Tvar a pravdepodobnosti nezakorenených evolučných stromov atómov nie sú závislé

len od histórie ale aj od toho, na ktorú sekvenciu atómov v histórii sa pozeráme. Ak by sme v súčasnej sekvencii mali päť atómov jedného typu označených A,B,C,D a E, môžu stromy týchto atómov vyzerať ako na obrázku 3.6. Pokiaľ však posledná udalosť bola duplikácia, ktorou vznikli atómy B a C, z pohľadu predošlej sekvencie uvažujeme už stromy ako na obrázku 3.7.

Pri spätnej rekonštrukcii duplikácie musíme náležite upraviť množinu evolučných stromov. Pre každú dvojicu atómov (x, y), ktorá duplikáciou vznikla, najprv zahodíme všetky stromy, v ktorých x a y nemali spoločného suseda. Aposteriórne pravdepodobnosti stromov, ktoré ostanú, preškálujeme, aby súčet ich pravdepodobností bol jedna. Následne zo stromov odstránime vrcholy x a y. Vrchol, ktorý sa takto stal novým listom, dostane nové označenie, ktoré vznikne zjednotením označení x a y. Tak, ako v podkapitole **??**, atóm identifikujeme množinou všetkých atómov súčasnej sekvencie, ktoré z neho vzniknú sériou jednej alebo viacerých duplikácií.

Na obrázku 3.8 môžeme vidieť množinu stromov, ktorá ostane po spojení atómov A a E.



$p_1 = 0.863$ $p_2 = 0.069$ $p_3 = 0.068$

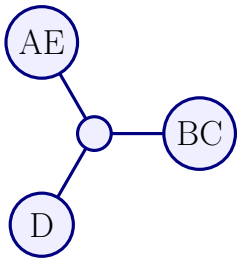
Obr. 3.7: Množina evolučných stromov, ktorá vznikne po spojení atómov B a C v stromoch na obrázku 3.6. Pre štyri atómy existujú vždy najviac tri rôzne stromy.

3.5 Čerešňovitosť

V súvislosti s evolučnými stromami atómov zavedieme nový pojem, ktorý nazývame čerešňovitosť.

Čerešňovitosť dvoch atómov rovnakého typu je odhadovaná aposteriórna pravdepodobnosť, že tieto dva atómy tvoria takzvanú čerešňu v evolučnom strome atómov. Dva atómy tvoria čerešňu, keď sú listami daného stromu a zároveň majú spoločného suseda.

Čerešňovitosť atómov x a y označujeme $c(x, y)$. Jej hodnotu vypočítame tak, že sčítame aposteriórne pravdepodobnosti evolučných stromov, v ktorých x a y tvoria



$$p_1 = 1$$

Obr. 3.8: Množina evolučných stromov, ktorá vznikne po spojení atómov A a E v stromoch na obrázku 3.7. Zahodili sme dva zo stromov, lebo A a E v nich nemali spoločného suseda. Pre ľubovoľné tri atómy je evolučný strom vždy len jeden.

čerešňu. Napríklad na obrázku 3.6, $c(B, C) = 1$, $c(A, D) = 0.069$, $c(A, B) = 0$.

Čerešňovitosť je definovaná len pre dvojice atómov, ktoré sa nachádzajú v aktuálnej sekvencii. Navyše hodnota čerešňovitosti závisí od sekvencie v ktorej sa atómy nachádzajú. Napríklad pre sekvencie, ktorým zodpovedajú stromy na obrázkoch 3.6 a 3.7, je $c(D, BC)$ rôzna. Pre pôvodnú sekvenciu a jej stromy na obrázku 3.5 nie je $c(D, BC)$ definovaná.

Pokiaľ máme len tri atómy jedného typu, čerešňovitosť ľubovoľnej dvojice je 1, pretože pre tri atómy existuje len jeden nezakorenený evolučný strom, v ktorom všetky dvojice listov tvoria čerešňu. Špeciálne dodefinujeme, že pokiaľ z nejakého typu máme len dva atómy, ich čerešňovitosť bude jedna. Stromy pre dva alebo tri atómy ani nebudeme konštruovať

Vysoká hodnota čerešňovitosti dvoch atómov ešte nemusí znamenať, že atómy vznikli spoločnou duplikáciou. Funguje to zvyčajne opačne, ak je hodnota čerešňovitosti nízka, máme dôvod pochybovať o duplikácii, ktorá by dané atómy vytvorila.

Výhodou čerešňovitosti je, že ju dokážeme počítať veľmi rýchlo, pokiaľ reprezentujeme stromy rozumným spôsobom. (V našej implementácii si pamätáme pre každý vrchol zoznam jeho susedov, takže overiť, či nejaké dva listy majú rovnakého suseda vieme v konštantnom čase). Časová zložitosť spočítania čerešňovitosti dvoch atómov je lineárna od počtu evolučných stromov pre daný typ atómov. Tých je obvykle málo, väčšinou do päť. Jediná zdĺhavejšia časť je, že pred prvou rekonštrukciou histórie musíme navzorkovať evolučné stromy pre atómy v súčasnej sekvencii. To nám však stačí spraviť raz (podobne, ako rozdelenie sekvencie na atómy).

Malou nevýhodou je, že čerešňovitosť nám začne výraznejšie pomáhať až pri zložitejších históriách. Pokiaľ napríklad máme z každého atómu tri alebo menej kópií, čerešňovitosť nám nedá žiadnu dodatočnú informáciu. Možno však pri týchto jednoduchších

históriách pomoc nepotrebujeme a/alebo aj tak nevieme získať užitočné informácie o príbuznosti atómov.

3.6 Využitie čerešňovitosti pri návrhu kandidátov

Vďaka tomu, že $c(a, b)$ závisí len od atómov a a b, dokážeme čerešňovitosť dobre využiť v algoritme, ktorý vzorkuje kandidátov na duplikačné udalosti. Stačí nám upraviť ceny hrán v grafe, ktorý sme v algoritme používali.

Ceny hrán vedúcich do vrcholov $(1, i, j)$ a $(4, i, j)$, ktoré mali pôvodnú cenu 2 vynásobíme hodnotou čerešňovitosti atómov na pozíciách i a j. Tým pádom sa cena cesty, a teda aj pravdepodobnosť výberu duplikácie, prenásobí súčinom čerešňovitostí dvojíc atómov, ktoré by duplikáciou vznikli.

Pokiaľ sekvenciu atómov, ktorú práve rekonštruujeme, označíme $a_1, a_2, \ldots a_n$ a pokiaľ i a j budú pozície začiatkov sekvencií, ktoré vzniknú duplikáciou u pozostávajúcej z ℓ atómov, tak skóre tejto duplikácie bude

$$\prod_{k=0}^{\ell-1} 2c(a_{i+k}, a_{j+k}) \tag{3.3}$$

3.7 Dôvera v čerešňovitosť

Aposteriórne pravdepodobnosti evolučných stromov sú len odhadované, čiže na hodnoty čerešňovitosti sa nemôžeme bezvýhradne spoliehať. Uvedieme preto dva spôsoby ako môžeme upraviť počítanie čerešňovitosti.

Vo veľkej väčšine prípadov nám program MrBayes navzorkuje len zopár rôznych evolučných stromov, no sem-tam sa stane, že DNA sekvencie atómov sú príliš „neurčité“ a správny evolučný strom je veľmi ťažké určiť. Napríklad nám môže MrBayes navzorkovať 80 stromov s aposteriórnymi pravdepodobnosťami od 0.03 po 0.001. V takýchto prípadoch je reálna šanca, že skutočne správny strom sa medzi navzorkovanými stromami ani nenachádza. Po niekoľkých zrekonštruovaných udalostiach by sme sa tak dostali do stavu, kedy v správnej udalosti majú nejaké atómy čerešňovitosť nula. Nebolo by dobré takto prísne penalizovať potenciálne dobré udalosti.

Preto zavedieme tzv. *opravenú čerešňovitosť*. Nech p_{max} je aposteriórna pravdepodobnosť najpravdepodobnejšieho evolučného stromu atómov typu t podľa programu MrBayes. Potom opravená čerešňovitosť atómov a a b typu t má hodnotu

$$c'(a, b) = min\{1, c(a, b) + (1 - \sqrt{p_{max}})^2\} \tag{3.4}$$

Napríklad pre stromy na obrázku 3.6, kde $p_{max} = 0.863$ by sme pripočítavali zanedbateľnú hodnotu 0.005. Keď je však v stromoch veľká neistota a napríklad $p_{max} = 0.03$, pripočítame až 0.684, čím dostaneme čerešňovitosti na približne rovnakú úroveň. Testovali sme aj iné opravné funkcie a táto najviac pomohla pri rekonštrukcii.

Čím sú DNA sekvencie atómov dlhšie, tým viac sa môžeme spoľahnúť na navzorkované evolučné stromy [VBSS10]. Preto sme v tejto práci skúšali brať menej do úvahy čerešňovitosť krátkych atómov. Keďže pravdepodobnosť navzorkovania udalosti závisí od súčinu čerešňovitostí, posunieme čerešňovitosť kratších atómov bližšie k jednotke. Nech ℓ je dĺžka DNA sekvencie kratšieho z atómov a, b (vplyvom lokálnych delécií a inzercií môžu mať atómy mierne odlišné dĺžky).

$$c''(a,b) = c'(a,b)^{\ell/(\ell+1000)} \tag{3.5}$$

V algoritme sme používali len $c'(a,b)$ a $c''(a,b)$, pričom porovnanie týchto metód nájdeme v podkapitole 5.3. Pokiaľ v nasledujúcom texte budeme používať pojem čerešňovitosť, ak neuvedieme inak, budeme mať namysli funkciu $c'(a,b)$.

Kapitola 4

Skórovanie udalostí

V tejto kapitole sa zaoberáme tým, ako spomedzi množiny kandidátov na duplikačné udalosti vybrať správnu výslednú udalosť.

Postupujeme tak, že pre každého kandidáta spočítame niekoľko reálnych čísel, tzv. *parametre kandidáta*. Parametre kandidáta u označujeme $s_1(u), \ldots, s_n(u)$. Medzi parametre môžeme zaradiť ľubovolné vlastnosti udalosti, ktoré by mohli indikovať, či je udalosť správna alebo nie. Z týchto parametrov napokon spočítame jedno výsledné reálne číslo, skóre kandidáta. Skóre kandidáta u označujeme $s(u)$.

Výsledného kandidáta spomedzi množiny kandidátov K vyberieme pravdepodobnostne, pričom pravdepodobnosť výberu kandidáta, ktorý zodpovedá udalosti u, je

$$P[u] = \frac{s(u)}{\sum_{u' \in K} s(u')} \tag{4.1}$$

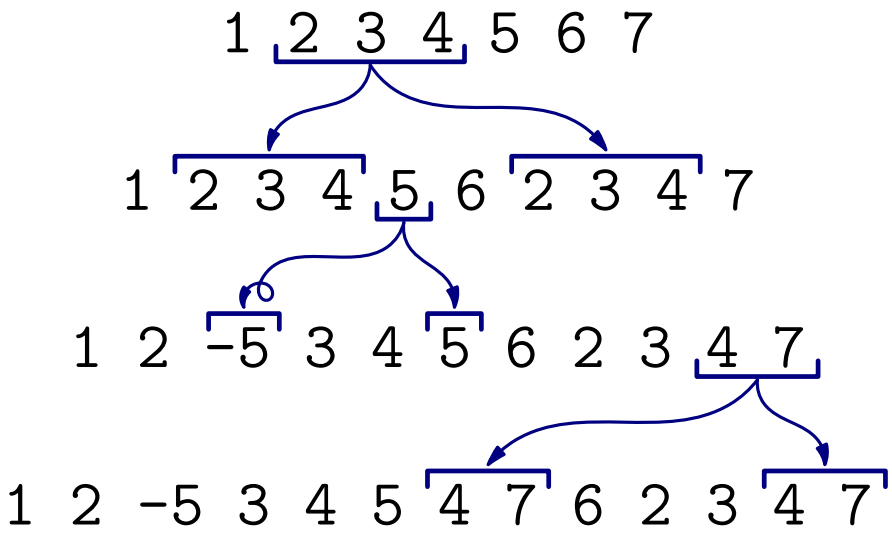
V podkapitolách 4.1 a 4.2 sa budeme podrobnejšie venovať parametrom kandidátov. V podkapitolách 4.3 a 4.4 zasa vysvetlíme, ako na základe parametrov spočítať skóre udalosti.

4.1 Susedné páry atómov a predĺžiteľnosť udalostí

V tejto podkapitole zadefinujeme a vysvetlíme pojmy *susedný pár*, *silný pár* a *predĺžiteľná duplikácia*, ktoré poslúžia ako základy niektorých dôležitých parametrov udalostí.

Pozrime sa lepšie na to, ako vyzerajú histórie sekvencií atómov. Pre ilustráciu uvedieme jednu konkrétnu históriu, ktorá začína sekvenciou `1 2 3 4 5 6 7` a v ktorej postupne nastanú tri duplikácie znázornené na obrázku 4.1. Pri bližšom pohľade na túto históriu si môžeme všimnúť, že dvojice po sebe idúcich čísel sa až tak nemenia. Na začiatku sme mali takéto dvojice: $(1, 2)$, $(2, 3)$, $(3, 4)$, $(4, 5)$, $(5, 6)$, $(6, 7)$. Po prvej udalosti sa väčšina dvojíc nezmenila, ubudla dvojica $(6, 7)$ a pribudli dvojice $(6, 2)$,

$(4, 7)$. Pri ďalšej udalosti už žiadna dvojica neubudla (pretože $(2, 3)$ sa nachádza aj na inom mieste), no pribudli dvojice $(2, -5)$ a $(-5, 3)$. Treťou udalosťou zanikla dvojica $(5, 6)$ a vznikli dve nové dvojice $(5, 4)$ a $(7, 6)$.



Obr. 4.1: Ukážkový príklad histórie, ktorá začala sekvenciou atómov `1 2 3 4 5 6 7`. V histórii najprv nastala duplikácia, potom duplikácia s inverziou a znova duplikácia.

Pri správnych históriách vidíme, že po udalosti obvykle vzniknú dve nové dvojice. Avšak kvôli tomu, že pri výpočte rozdelenia DNA sekvencie na atómy sa obvykle stratia krátke atómy (viď podkapitolu 2.2), môže sa ľahko stať, že vznikne len jedna nová dvojica, prípadne žiadna.

Definícia 4.1 (Susedné páry)**.** Majme sekvenciu atómov $a_1, a_2, \ldots a_n$. Definujme $b_0 \ldots b_{n+1}$ ako postupnosť, kde $b_0 = ¢$, $b_{n+1} = \$$ a pre $\forall i \in \{1..n\}$, b_i je číslo typu atómu a_i vynásobené -1, ak je atóm a_i invertovaný. ¢ a \$ sú špeciálne znaky, ktoré označujú začiatok a koniec sekvencie.

Množina *susedných párov* sekvencie $a_1, \ldots, a_n$ je množina usporiadaných dvojíc (b_i, b_{i+1}) a $(-b_{i+1}, -b_i)$, pre $i \in \{0..n\}$.

Všimnime si, že okraje sekvencie majú podľa definície tiež zodpovedajúce susedné páry. Napríklad v sekvenciách na obrázku 4.1 máme susedné páry $(¢, 1)$, $(-1, ¢)$, $(7, \$)$, $(\$, -7)$. Kvôli inverzným duplikáciám sme definovali susedné páry tak, aby (x, y) bol susedný pár práve vtedy, keď $(-y, -x)$ je susedný pár. Vďaka tomu, ak duplikujeme `1 2 3 4`, na `-4 -3 -2 -1` a budeme sledovať ako sa zmení množina susedných párov, nepribudnú páry $(-4, -3)$, $(-3, -2)$, $(-2, -1)$, ktoré tam už podľa definície boli. Ďalej môžeme považovať (x, y) a $(-y, -x)$ za ten istý susedný pár.

Videli sme tiež, že niekedy po udalosti jeden susedný pár zanikne a niekedy nie. Ak však chceme sledovať zanikanie dvojíc, je lepšie pozrieť sa na podtriedu susedných párov, ktorú nazývame *silné páry*.

Definícia 4.2 (Silné páry)**.** Nech M je množina susedných párov sekvencie S. Potom *silný pár* je dvojica (x, y) taká, že $(x, y) \in M$ a zároveň $\forall z \neq y : (x, z) \notin M$ a tiež $\forall z \neq x : (z, y) \notin M$. Slovne, (x, y) je silný pár, ak x má jednoznačne určeného nasledovníka a y má jednoznačne určeného predchodcu.

Všimnime si, že (x, y) je silný pár práve vtedy, keď $(-y, -x)$ je silný pár. Všimnime si tiež, že keď je v sekvencii najviac jeden atóm každého typu, každý susedný pár je zároveň silný pár.

Tak je to aj na začiatku histórie z obrázku 4.1. Na tejto histórii navyše môžeme vidieť rozdiel medzi silnými a susednými pármi. Napríklad po druhej udalosti páry $(5, 6)$ a $(2, 3)$ prestali byť silnými pármi a žiadne silné páry nevznikli. Naopak nezanikol žiaden susedný pár a vnikli dva nové susedné páry.

Počet susedných resp. silných párov, ktoré udalosť vytvorí, resp. odstráni, považujeme za dôležité vlastnosti, ktoré môžu napovedať, či je udalosť správna alebo nie. Preto sme tieto počty zaradili medzi parametre udalosti.

Keď nastane duplikácia atómov $a_1, a_2, \ldots, a_\ell$, tak vzniknú v sekvencii dva opakujúce sa úseky dlhé ℓ (alebo viac) atómov. Všimnime si však, že ak poznáme len sekvenciu ktorá vznikla a vidíme len opakujúce sa úseky, nemôžeme s istotou vylúčiť žiadnu z potenciálnych udalostí, ktoré by kopírovali len časť týchto atómov. Pre ľubovoľné $1 \leq i \leq j \leq \ell$ mohla teoreticky nastať udalosť, ktorá kopírovala len atómy $a_i, \ldots, a_j$. A práve pri vzorkovaní kandidátov (viď kapitolu 3) sa takéto nekompletné udalosti dostávajú medzi navzorkovaných kandidátov. Celková pravdepodobnosť výberu všetkých kratších udalostí dokopy je dokonca väčšia alebo rovná ako pravdepodobnosť výberu tej najdlhšej (ak $\ell > 1$).

(Pre zdôvodnenie tohto tvrdenia sa stačí pozrieť na udalosti dĺžky $\ell - 1$. Podľa vzťahu 3.3 pravdepodobnosť výberu každej z týchto čiastočných udalostí je aspoň polovica pravdepodobnosti výberu celej udalosti dĺžky ℓ. Využívame, že $c(a, b)$ v tomto vzťahu je menšie alebo rovné 1.)

Obvykle však takíto kratší kandidáti nie sú správni a aby sme sa ich dokázali zbavovať, budeme sledovať takzvanú *predĺžiteľnosť duplikácie*. Pokiaľ by predĺžením duplikácie vznikla neplatná udalosť (t.j. predĺžené úseky by sa prekrývali), tak pôvodnú udalosť nepovažujeme za predĺžiteľnú, čo je obsiahnuté aj v definícii. Pripomíname, že orientovaný typ je typ atómu aj so znamienkom.

Definícia 4.3 (Predĺžiteľné duplikácie)**.** Duplikácia bez inverzie, ktorá v sekvencii $a_1, a_2, \ldots, a_n$ skopíruje atómy a_i až a_j za atóm a_k je *zľava predĺžiteľná*, ak $k \neq j$, $k \neq i-1$ a a_{i-1} má rovnaký orientovaný typ, ako a_k.

Duplikácia bez inverzie, ktorá v sekvencii $a_1, a_2, \ldots, a_n$ skopíruje atómy a_i až a_j pred atóm a_k je *sprava predĺžiteľná*, ak $k \neq i$, $k \neq j+1$ a a_{j+1} má rovnaký orientovaný typ, ako a_k.

Duplikácia s inverziou, ktorá v sekvencii $a_1, a_2, \ldots, a_n$ skopíruje atómy a_i až a_j pred atóm a_k je *zľava predĺžiteľná*, ak $k \neq i$, $k \neq i-1$ a a_{i-1} má opačný orientovaný typ, ako a_k.

Duplikácia s inverziou, ktorá v sekvencii $a_1, a_2, \ldots, a_n$ skopíruje atómy a_i až a_j za atóm a_k je *sprava predĺžiteľná*, ak $k \neq j$, $k \neq j+1$ a a_{j+1} má opačný orientovaný typ, ako a_k.

4.2 Zoznam parametrov udalostí

Duplikačnú udalosť, ktorej parametre nás zaujímajú, označíme u. Táto udalosť sa skladá z jednej duplikácie (s prípadnou inverziou) a niekoľkých (väčšinou nula) nasledujúcich delécií. Označme si sekvenciu atómov, ktorá je v histórii pred vykonaním duplikácie, S_a. Sekvenciu, ktorá vznikne po vykonaní duplikácie a všetkých delécií, si označme S_b. Sekvenciu S_b poznáme a z danej sekvencie S_b a udalosti u vieme sekvenciu S_a ľahko spočítať.

Vlastnosti takto popísanej udalosti vyjadríme pomocou sady parametrov. Každý parameter zachytáva nejakú vlastnosť alebo kombináciu vlastností udalosti. V tabuľke 4.1 sú vymenované a stručne popísané všetky parametere, ktoré sme v tejto práci používali.

Parametre s_1 až s_3 nie sú priamo vlastnosti udalosti, ale slúžia nám na to, aby sme mohli vyjadriť napríklad pomer dĺžky udalosti k celkovej dĺžke sekvencie. Namiesto pôvodných číselných vlastností S_b používame logaritmy jednak preto, že potom môžeme namiesto podielov počítať rozdiely, a tiež preto, že potom nemajú tieto parametre príliš veľké hodnoty v porovnaní s inými parametrami.

Parametre s_4 až s_7 reprezentujú základné vlastnosti duplikačnej udalosti, medzi ktoré patrí počet duplikovaných atómov (dĺžka duplikácie), vzdialenosť, počet delécií a celková dĺžka delécií, ktoré nastanú po duplikácii. Vzdialenosť duplikácie je počet atómov medzi úsekmi sekvencie, ktoré vznikli duplikáciou. Napríklad, ak postupnosť `(1 2) 3 4 5 (1 2)` vznikla duplikáciou atómov `1 2`, tak vzdialenosť tejto duplikácie je 3.

Na základe podkapitoly 4.1 sme pridali parametre s_8 až s_{12} popisujúce predĺžiteľnosť

s_1	Logaritmus dĺžky S_b
s_2	Logaritmus počtu rôznych typov atómov v S_b
s_3	Logaritmus počtu rôznych susedných párov v S_b
s_4	Logaritmus dĺžky duplikácie
s_5	$\log(1+v)$, kde v je vzdialenosť duplikácie
s_6	Počet delécií v duplikačnej udalosti u
s_7	Logaritmus z celkového počtu atómov v deléciách v u
s_8	2, ak je duplikácia predĺžiteľná do oboch strán 1, ak je duplikácia predĺžiteľná do jednej strany 0, ak duplikácia nie je predĺžiteľná
s_9	Počet silných párov, ktoré sú v S_a, ale nie sú v S_b
s_{10}	Počet silných párov, ktoré sú v S_b, ale nie sú v S_a
s_{11}	Počet susedných párov, ktoré sú v S_a, ale nie sú v S_b
s_{12}	Počet susedných párov, ktoré sú v S_b, ale nie sú v S_a
s_{13}	0, pre duplikáciu bez inverzie $\log(\frac{1+b}{1+a}) - 0.5$, pre duplikáciu s inverziou, kde a je počet atómov, ktoré sa v S_a vyskytujú s oboma znamienkami a b je počet takých atómov v S_b.
s_{14}	Priemerná čerešňovitosť duplikovaných atómov
s_{15}	Súčin čerešňovitostí duplikovaných atómov
s_{16}	Súčin čerešňovitostí, pričom menej berieme do úvahy atómy s krátkou dĺžkou DNA (podkapitola 3.7)
$s_{0,i}$	e^{s_i} pre $1 \leq i \leq 16$
$s_{1,i,j}$	$\frac{s_i}{1+s_j}$ pre $1 \leq i \leq 16$ a $1 \leq j \leq 4$
$s_{2,i,j}$	$s_i \cdot s_j$ pre $1 \leq i \leq 16$ a $1 \leq j \leq 4$

Tabuľka 4.1: Tabuľka základných a odvodených parametrov duplikačnej udalosti, na základe ktorých počítame skóre udalosti. Celkový počet parametrov je 160.

duplikácie, počty susedných párov a počty silných párov, pričom (x, y) a $(-y, -x)$ počítame ako jeden pár.

Ďalší parameter s_{13} sa snaží zachytiť nasledujúcu skutočnosť. Na začiatku evolúcie sa každý atóm vyskytuje len s kladným alebo len so záporným znamienkom. Inverzná duplikácia obvykle zvýši počet atómov, ktoré sa vyskytujú s oboma znamienkami, najmä vtedy, ak bolo obojznamienkových atómov málo. Preto prirodzene od inverznej duplikácie očakávame, že v S_a bude menej obojznamienkových atómov ako v S_b. Najmä vtedy, ak ich v S_a bolo málo.

Posledné tri základné parametre s_{14} až s_{16} zachytávajú čerešňovitosť, ktorú popisujeme v podkapitole 3.5. Medzi parametre sme začlenili súčet a dva varianty súčinu čerešňovitostí dvojíc atómov, ktoré vzniknú duplikáciou.

Okrem 16 základných parametrov sme pridali aj niekoľko odvodených. Odvodené parametre vznikli buď ako súčin dvoch základných alebo ako podiel dvoch základných alebo ako e umocnené na nejaký základný parameter. Uvažovali sme len súčiny a podiely s niektorým z parametrov s_1, s_2, s_3, s_4. Celkovo sme teda mali 160 parametrov, pričom v kapitole 5 experimentálne overíme, či pridaných 144 parametrov malo pozitívny vplyv na rekonštrukciu.

4.3 Výpočet celkového skóre

Vidíme, že parametrov udalosti je pomerne veľa. Bežnými metódami by bolo náročné určiť, ako z nich čo najlepšie vypočítať celkové skóre.

V predošlej práci [Hoz14], v ktorej sme sa tiež zaoberali skórovaním udalostí, sme mali len šesť parametrov pre každú udalosť. Jeden z týchto parametrov bol ekvivalentný nášmu parametru s_4 (logaritmus dĺžky duplikácie), jeden nášmu parametru s_8 (predĺžiteľnosť duplikácie) a jeden súvisel s počtom susedných párov (hodnota tohto tretieho parametra bola $s_{12} - s_{11} + s_{13}$). Zvyšné tri parametre nie sú pre našu prácu relevantné, takže ich nebudeme popisovať.

Každý z týchto parametrov s'_1 až s'_6 bol vynásobený jedným zo šiestich koeficientov k'_1 až k'_6, podľa toho, ako veľmi mal ovplyvňovať celkové skóre. To bolo dané vzťahom

$$s' = e^{-d+\sum_{i=1}^{6} s'_i \cdot k'_i} \tag{4.2}$$

Hodnota d vo vzťahu vyjadrovala počet delécií. Koeficienty sme v predchádzajúcej práci určovali empiricky, na základe série experimentov. Najprv sme menili hodnotu jedného koeficientu (ostatné boli nastavené na nulu) a sledovali, ako sa mení vierohodnosť histórií, ktoré vznikali rekonštrukciou s takýmto skórovaním. Následne sme testovali rôzne kombinácie všetkých koeficientov, až sme dosiahli také nastavenie, s ktorým sme boli spokojní. Jednou z víťazných kombinácii koeficientov bolo $(k'_1, k'_2, k'_3, k'_4, k'_5, k'_6) = (1, 0, -2, 6, 0, 0)$, pričom nulové koeficienty boli pri tých parametroch, ktoré nie sú pre našu prácu relevantné.

Pri použití našich parametrov, by sme skóre z predošlej práce vypočítali vzťahom

$$s' = e^{s_4 - s_6 - 2s_8 - 6s_{11} + 6s_{11} + 6s_{13}} \tag{4.3}$$

Aj túto skórovaciu metódu porovnáme v kapitole 5 so skórovacou metódou z našej

práce.

V tejto práci na vypočítanie skóre z množstva parametrov využijeme metódy strojového učenia, konkrétne učenie s učiteľom (angl. supervised learning). Aby sme však mohli použiť takýto prístup, potrebujeme mať trénovacie dáta, čiže dáta, v ktorých poznáme hodnoty nielen všetkých parametrov pre danú udalosť, ale poznáme aj správnu hodnotu celkového skóre. V ideálnom prípade majú správne udalosti skóre 1 a nesprávne udalosti skóre 0.

Naše trénovacie dáta budú mať tvar $(s_1, s_2, \ldots, s_n, y)$, pričom $y = 1$, pokiaľ udalosť s parametrami $s_1, \ldots, s_n$ je správna a $y = 0$ v opačnom prípade.

Aby sme dokázali posúdiť správnosť udalosti, budeme vytvárať trénovacie dáta na základe simulovaných histórií, v ktorých správne udalosti poznáme. Z jednej simulovanej histórie dostaneme niekoľko trénovacích dát. Pozrieme sa najprv na poslednú sekvenciu v histórii a algoritmom z kapitoly 3 vygenerujeme kandidátov na poslednú udalosť. Rozdelíme kandidátov na pozitívne prípady (kedy kandidát zodpovedá správnej udalosti podľa histórie) a negatívne prípady (kedy kandidát nezodpovedá správnej udalosti). Nech počty týchto prípadov sú a a b. Medzi trénovacie dáta dáme náhodných $\min(a, b)$ pozitívnych prípadov a náhodných $\min(a, b)$ negatívnych prípadov. Pozitívnych aj negatívnych trénovacích dát bude tým pádom rovnako veľa. Pôvodne, keď sme do trénovacích dát dávali úplne všetkých kandidátov, boli tieto dáta veľmi nevyvážené a trénovací algoritmus mal problémy s učením. Proces opakujeme s históriou bez poslednej udalosti, potom bez predposlednej, atď., až kým nedostaneme históriu bez duplikácií.

Na týchto trénovacích dátach môžeme natrénovať model logistickej regresie, ktorý bude následne schopný predikovať, či je udalosť správna na základe jej parametrov. Predikovaná hodnota (skóre udalosti) už nebude len z $\{0, 1\}$, ale môže byť hocikde v intervale $(0, 1)$, podľa toho, ako veľmi si je model istý, že je udalosť správna resp. nesprávna. Ak náš model natrénujeme dobre, skóre správnych udalostí bude blízke jednotke a skóre nesprávnych udalostí bude blízke nule.

Musíme sa však vysporiadať s jedným problémom. V podkapitole 2.5 sme uviedli dva spôsoby určovania toho, či je udalosť správna alebo nie. Udalosť je striktne správna, pokiaľ je správna a zároveň má správny smer. Pokiaľ všetky správne udalosti, teda aj tie, ktoré nie sú striktne správne, budeme považovať za pozitívne trénovacie dáta, algoritmus sa medzi nimi nenaučí rozlišovať. V niektorých prípadoch (viď. obrázok 2.4) je však potrebné pre správnu rekonštrukciu histórie uhádnuť aj správny smer. Pokiaľ náš algoritmus nie je natrénovaný na rozlišovanie smeru, bude pravdepodobnosť, že uhádne striktne správnu udalosť najviac jedna polovica. Pravdepodobnosť, že správne zrekonštruujeme históriu, v ktorej pri k udalostiach záleží na smere je najviac 2^{-k}.

Ak by nám takéto obmedzenie prekážalo, môžeme algoritmus natrénovať, aby rozlišoval aj smer udalostí. Teda pozitívne trénovacie dáta budú len striktne správne udalosti a ostatné dáta budú negatívne. Pri tomto prístupe však musíme počítať s tým, že sa v dátach môžu vyskytnúť dve udalosti s rovnakými parametrami, ale raz s $y = 0$ a raz s $y = 1$, čo by sa mohlo stať napríklad pre udalosť na obrázku 2.3.

V kapitole 5 porovnáme, ktorý z týchto prístupov funguje lepšie.

4.4 Logistická regresia

Logistická regresia je štatistická metóda používaná na hľadanie závislostí medzi vstupnými parametrami, ktoré sú reprezentované ako mnohorozmerný vektor reálnych čísel, a výstupnou binárnou premennou [WD67]. Vstupné parametre označíme $s_1, \ldots, s_n$ a výstupnú premennú y.

Úlohou logistickej regresie je predikovať pravdepodobnosť, že $y = 1$. Na predikciu sa používa takzvaná logistická funkcia

$$P[y = 1] = L(k_0 + \sum_{i=1}^{n} k_i \cdot s_i) \tag{4.4}$$

pričom $k_0, \ldots, k_n$ sú takzvané koeficienty logistickej regresie a

$$L(x) = \frac{1}{1 + e^{-x}} \tag{4.5}$$

Funkcia $L(x)$ sa nazýva sigmoida a ide o špeciálny prípad logistickej funkcie. Je to rastúca nelineárna funkcia, ktorá mapuje hodnoty z $\mathbb{R}$ do intervalu $(0, 1)$.

Našim cieľom je nájsť koeficienty $k_0, \ldots k_n$, ktoré minimalizujú trénovaciu chybu $E(X, Y, k)$, kde X je postupnosť vstupných vektorov, $X_i = (1, s_{i,1}, \ldots, s_{i,n})$, Y je postupnosť k nim priradených hodnôt (1 alebo 0, podľa toho, či bola udalosť s parametrami $s_{i,1}, \ldots, s_{i,n}$ správna) a k je vektor koeficientov $k_0, \ldots k_n$.

$$E(X, Y, k) = \frac{1}{2} k^T k + \sum_{i=1}^{|X|} \log(1 + e^{-Y_i (X_i^T k)}) \tag{4.6}$$

Prvý člen $\frac{1}{2} k^T k$ sa nazýva L_2 regularizácia a zabraňuje, aby boli koeficienty zbytočne veľké, čo pomáha proti následnému zlyhaniu predikcie na testovacích dátach.

Na riešenie lineárnej regresie a nájdenie koeficientov pre naše dáta sme použili knižnicu scikit-learn (`http://scikit-learn.org`). Skóre udalosti určíme na základe predikovanej pravdepodobnosti.

$$s(u) = L(k_0 + \sum_{i=1}^{n} k_i \cdot s_i(u)) \tag{4.7}$$

4.5 Generovanie simulovaných histórií

Pre účely trénovania predikčného modelu a tiež kvôli tomu, aby sme mohli v kapitole 5 preskúmať a vyhodnotiť úspešnosť algoritmu, potrebujeme vygenerovať množstvo histórií, ktoré sa podobajú na skutočné histórie organizmov.

Začneme s tým, že vygenerujeme náhodnú DNA sekvenciu dlhú 100 000 báz, čo je podobná dĺžka, ako majú génové zhluky, ktoré skúmame. Túto sekvenciu necháme vyvíjať sa čas $t_{gen} = 0.04$ našich jednotiek času, čo tiež korešponduje s našimi reálnymi dátami (viď podkapitolu 1.4).

Pri generovaní histórií používame rovnaký pravdepodobnostný model, ako v predošlej práci [Hoz14], ktorý aproximuje evolúciu skutočných sekvencií.

Dlhé udalosti (duplikácie, duplikácie s inverziou a delécie) modelujeme Poissonovym procesom. Pri takomto procese je čas medzi dvoma udalosťami náhodná premenná s exponenciálnou distribúciou s parametrom λ. Parameter λ hovorí o tom, koľko udalostí sa priemerne udeje za jednotku času. Udalosti teda generujeme tak, že začneme v čase $t_0 = 0$ a náhodne určíme t_1, čas prvej udalosti podľa spomínanej distribúcie. Nasimulujeme lokálne mutácie až po čas t_1 a vyrobíme náhodnú udalosť.

Náhodná udalosť je s pravdepodobnosťou p_{del} delécia Ak udalosť nie je delécia, je to s pravdepodobnosťou p_{inv} duplikácia s inverziou. Inak je to duplikácia bez inverzie.

Dĺžka udalosti (dĺžka DNA sekvencie) je daná geometrickou distribúciou so strednou hodnotou m_ℓ. Pokiaľ by dĺžka udalosti presiahla dĺžku celej sekvencie, vygenerujeme dĺžku znova. Pozícia začiatku udalosti je vybraná rovnomerne náhodne z množiny $\{0, 1, \ldots, d\}$, kde d je rozdiel aktuálnej dĺžky sekvencie a dĺžky udalosti, t.j. tak, aby sa celý kopírovaný úsek nachádzal v sekvencii.

Ak išlo o duplikáciou (s inverziou alebo bez inverzie) vygenerujeme náhodnú vzdialenosť duplikovaných úsekov z geometrického rozdelenia s priemerom m_{vzd}. Náhodne s 50% pravdepodobnosťou je smer duplikácie zľava doprava a s 50% pravdepodobnosťou naopak. Pokiaľ by duplikovaný úsek mal byť mimo sekvencie, vygenerujeme vzdialenosť a smer znova. Hodnoty parametrov nášho modelu sú dané tabuľkou 4.2.

parameter	**hodnota**
λ	200
p_{del}	0.05
p_{inv}	0.39
m_ℓ	14 307
m_v	306 718

Tabuľka 4.2: Parametre nášho modelu.

Po vytvorení udalosti pokračujeme s ďalšou. Náhodne určíme jej čas, vlastnosti a nasimulujeme lokálne mutácie po udalosť. Toto opakujeme, kým čas presiahne hodnotu t_{gen}. V takom prípade ukončíme generovanie histórie a nasimulujeme lokálne mutácie len po čas t_{gen}, ktorý zodpovedá súčasnosti.

Lokálne mutácie simulujeme po blokoch. Vždy, keď sa posúvame o čas t, spočítame pre každú pozíciu v sekvencii, aká je pravdepodobnosť, že na danej pozícii za čas t od poslednej udalosti nastala substitúcia.

Túto pravdepodobnosť určíme na základe Jukes-Cantorovho modelu [JC69]. Podľa tohto modelu je pravdepodobnosť mutácie bázy x na bázu y rovnaká pre všetky dvojice (x, y), tiež táto pravdepodobnosť nezávisí od pozície bázy v sekvencii. Keďže jednotku času sme definovali ako priemerný čas za ktorý sa udeje jedna substitúcia v prepočte na jednu bázu (viď podkapitolu 1.4, je pravdepodobnosť, že sa za čas t zmení báza na konkrétnej pozícii v sekvencii, nasledovná

$$p(t) = \frac{3}{4}(1 - e^{-4t/3}) \tag{4.8}$$

V našom modeli z implementačných dôvodov neuvažujeme krátke inzercie, ale len krátke delécie. Výsledný efekt je približne rovnaký a významne to zjednodušilo implementáciu generujúceho algoritmu. Predpokladáme, že generovať len krátke delécie, je dostatočnou aproximáciou skutočnej evolúcie, pre potreby ich rekonštrukcie.

Kapitola 5

Experimentálne výsledky

5.1 Doladenie parametrov

V samotnom algoritme je niekoľko parametrov a konštánt, ktorých hodnoty nevieme teoreticky podložiť. Napríklad pri generovaní kandidátov v podkapitole **??** vzorkujeme 25% kandidátov funkciou $1.2^{\ell-1} \prod_{k=1}^{\ell} c(a_{i+k}, a_{j+k})$ a zvyšných 75% funkciou $2^{\ell-1} \prod_{k=1}^{\ell} c(a_{i+k}, a_{j+k})$. Prečo práve pomer 1 : 3 a prečo práve koeficient 1.2? Ďalším príkladom je vyvažovacia hodnota pri počítaní čerešňovitosti, ktorú popisujeme v podkapitole **??**. Prečo by funkcia $(1 - \sqrt{x})^2$ mala byť najlepšia?

Takéto hodnoty a funkcie sme určovali empiricko-intuitívne. Vyskúšali sme niekoľko rôznych možností a sledovali sme, ako dobre sa algoritmu darí vzorkovať kandidátov. Na základe tohto sme vybrali hodnoty, ktoré sa nám zdali najlepšie. Dôležité bolo nastaviť tieto parametre na zmysluplné hodnoty a to, už či je koeficient 1.2 alebo 1.25, nie je podľa nás z praktického hľadiska dôležité.

Aby sme sa však vyhli tomu efektu, že náš algoritmus dáva dobre výsledky, len kvôli tomu, že je *nastavený* na našu množinu simulovaných histórií, po tom, ako sme nastavili všetky parametre sme nanovo vygenerovali nové simulované histórie, a až s týmito históriami sme robili merania, ktoré uvedieme v nasledujúcich podkapitolách.

5.2 Charakteristiky simulovaných histórií

Aby sme ohodnotili úspešnosť nášho algoritmu a tiež vplyv jednotlivých vylepšení a nových prístupov na úspešnú rekonštrukciu, vygenerovali sme 100 simulovaných histórií. Tieto histórie mali rôzne počty udalostí, od jednej až po pätnásť. Niektoré histórie boli v istom zmysle ťažšie, niektoré ľahšie.

Vo väčšine výsledkov experimentov preto uvádzame nielen priemerné výsledky ale aj desiatu najmenšiu hodnotu, desiatu najväčšiu hodnotu a medián.

Navyše, aby sme mohli sledovať, ako sa mení obtiažnosť rekonštrukcie od počtu udalostí, vybrali sme 5 ukážkových histórií.

Pre každé $m \in \{3, 6, 9, 12, 15\}$ sme vybrali náhodnú históriu z tých, ktoré mali presne m udalostí a tieto simulované histórie sme postupne označili $S_3, S_6, S_9, S_{12}, S_{15}$.

Štatistické údaje o simulovaných históriách môžeme vidieť v tabuľke 5.1.

	avg	min	max	S_3	S_6	S_9	S_{12}	S_{15}
Počet udalostí	8.17	1	15	3	6	9	12	15
Počet atómov	73.11	5	247	17	49	64	92	247
Počet typov atómov	24.53	4	44	10	19	26	34	44

Tabuľka 5.1: Vlastnosti simulovaných histórií. avg označuje priemer zo všetkých 100 hodnôt, min najmenšiu hodnotu a max najväčšiu. Ostatné stĺpce patria vybraných históriám $S_3, S_6, \ldots, S_{15}$

V nasledujúcich podkapitolách dôkladne otestujeme jednotlivé časti algoritmu. Najprv sa pozrieme na generovanie kandidátov, ktoré sme popisovali v kapitole 3. Potom sa pozrieme, ako dobre dokážeme z kandidátov vybrať jednu udalosť podľa algoritmu z kapitoly 4. Následne sa pozrieme na samotné rekonštrukcie histórií.

5.3 Generovanie kandidátov

V tejto podkapitole porovnáme tri spôsoby generovania kandidátov na duplikačné udalosti.

Prvý spôsob je pôvodná verzia vzorkovacieho algoritmu, bez použitia čerešňovitosti. Druhý spôsob je algoritmus, ktorý pri vzorkovaní využíva čerešňovitosť danú vzťahom 3.4. Tretí spôsob tiež využíva čerešňovitosť, avšak berie menej do úvahy čerešňovitosť atómov s krátkymi DNA sekvenciami. Tento tretí algoritmus na výpočet čerešňovitosti používa funkciu $c''(a, b)$ definovanú vzťahom 3.5. Označme tieto tri vzorkovacie algoritmy postupne A_0,A_1,A_2.

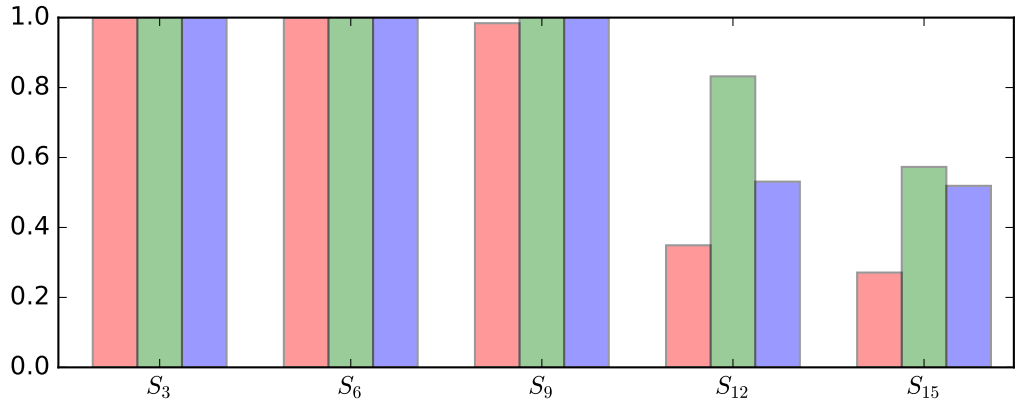
Porovnáme aj dva počty kandidátov, v jednom prípade necháme algoritmy navzorkovať $4n$ a v druhom prípade $2n$ kandidátov (n je počet atómov v sekvencii). Algoritmy budeme hodnotiť na základe toho, či sa im podarí do množiny kandidátov pre poslednú sekvenciu navzorkovať striktne správnu udalosť. V tomto prípade je však jedno, či hovoríme o správnej alebo striktne správnej udalosti, pretože v množine kandidátov je každá duplikačná udalosť buď s oboma smermi alebo s ani jedným.

Pre každý algoritmus A a každú históriu H nás zaujíma pravdepodobnosť $p_{A,H}$, že v množine kandidátov (navzorkovaných algoritmom A pre súčasnú sekvenciu histó-

rie H) sa bude nachádzať nejaká správna udalosť. Túto pravdepodobnosť odhadneme priemerom z 1 000 pokusov. V tabuľke 5.2 môžeme vidieť štatistiky hodnôt $p_{A,H}$ pre všetkých 100 histórií. Na obrázku 5.1 môžeme vidieť hodnoty $p_{A,H}$ pre päť konkrétnych histórií, a pre $4n$ navzorkovaných kandidátov.

Algoritmus	k	K_{avg}	p_{avg}	p_{10}	p_{50}	p_{90}
A_0	$4n$	189.9	0.85525	0.349	0.999	1.000
A_1	$4n$	162.0	0.96577	0.832	1.000	1.000
A_2	$4n$	171.8	0.95030	0.801	1.000	1.000
A_0	$2n$	123.3	0.78442	0.201	0.969	1.000
A_1	$2n$	114.4	0.92847	0.623	1.000	1.000
A_2	$2n$	118.4	0.91825	0.531	1.000	1.000

Tabuľka 5.2: k je počet navzorkovaných kandidátov, K_{avg} je priemerná veľkosť množiny kandidátov (po nakopírovaní každého kandidáta v oboch smeroch a odstránení duplikátov). p_{avg}, p_{10}, p_{50}, p_{90} sú postupne priemer, 10. percentil, medián a 90. percentil z hodnôt $p_{A,H}$ pre všetky simulované histórie.



Obr. 5.1: Pravdepodobnosť úspechu algoritmov A_0 (červený stĺpec), A_1 (zelený stĺpec) a A_2 (modrý stĺpec) pri vzorkovaní kandidátov vo vybraných súčasných sekvenciách.

Na základe výsledkov usudzujeme, že $4n$ kandidátov je primeraný počet. Pri $2n$ kandidátoch sa v ťažších históriách už príliš často stáva, že nenavzorkujeme správnu udalosť. Navyše veľkosť množiny kandidátov nie je až o toľko menšia, aby sa to vyplatilo.

Vidíme, že čerešňovitosť naozaj významne pomáha pri výbere kandidátov, najmä

pri ťažších históriách. Zaujímavé však je, že algoritmus A_2 nebol až taký úspešný a teda brať menší ohľad na čerešňovitosť kratších atómov nepomáha.

V tabuľke to síce nie je vidieť, ale aj pri jednoduchších históriách je algoritmus A_1 lepší ako zvyšné dva. Pri rovnakej hodnote p mal algoritmus A_1 väčšinou najmenšiu množinu kandidátov, niekedy vyrobil až dvakrát menej kandidátov ako algoritmus A_0. Pri ťažších históriách mali väčšinou A_1 a A_2 mierne väčšie množiny ako A_0 ale vynahradili to lepšou hodnotou p.

Veľkosť množiny kandidátov ovplyvňuje ďalšiu fázu algoritmu, v ktorej z kandidátov vyberáme jednu výslednú udalosť. Mať menej kandidátov je lepšie, lebo sa nám bude ľahšie vyberať udalosť v ďalšej fáze algoritmu.

Jednoznačný víťaz spomedzi algoritmov na generovanie kandidátov je A_1. V ďalších experimentoch, pokiaľ neuvedieme inak, budeme na generovanie kandidátov používať tento algoritmus s počtom navzorkovaných kandidátov $4n$.

5.4 Trénovanie modelu logistickej regresie

Ako sme vysvetlili v podkapitole 4.3 je dôležité, či trénovací model naučíme rozlišovať medzi udalosťami, ktoré vedú opačným smerom. Vo *voľných* trénovacích dátach sú všetky správne udalosti označené jednotkou. V *striktných* dátach sú pozitívne dáta len striktne správne udalosti.

Dáta pre logistickú regresiu sme získali z $5,000$ simulovaných histórií.

V prípade voľnejšej definície správnosti udalostí sme mali 203 676 dát. Tieto sme rozdelili na množinu T_1, ktorá slúžili na natrénovanie logistickej regresie a množinu T_2, ktorá slúžili na výpočet testovacej chyby v tabuľke 5.3. T_2 tvorilo 5 000 náhodne vybratých dát a T_1 tvorili zvyšné dáta.

V prípade striktnej definície správnosti udalosti sme mali 102 140 dát, ktoré sme analogicky rozdelili na T_3 a T_4. Opäť bolo 5 000 náhodne vybratých v T_4 a zvyšok bol v T_3.

Tento model sme trénovali pre 16 základných parametrov s_1 až s_{16} aj pre celú sadu 160 parametrov. Z hodnôt v tabuľkách 5.3 a 5.5 usudzujeme, že odvodené parametre nemajú významný vplyv na úspešnosť algoritmu.

V tabuľke 5.4 uvádzame koeficienty natrénovaných modelov logistickej regresie pre 16 parametrov. Skutočným prekvapením bolo to, že pri striktne hodnotených udalostiach bola hodnota k_{14} – koeficient pri parametri, ktorý vyjadruje priemernú čerešňovitosť – záporná, akoby udalosti s vysokou priemernou čerešňovitosťou neboli správne.

Trénovacie dáta	Počet parametrov	P	P_{T_2}	P_{T_4}
T_1	16	0.939	0.936	0.918
T_3	16	0.958	0.825	0.966
T_1	160	0.947	0.951	0.928
T_3	160	0.959	0.839	0.963

Tabuľka 5.3: Úspešnosť predikcie logistickej regresie. P je počet správne predikovaných hodnôt na trénovacej množine, P_{T_2} je počet správne predikovaných hodnôt na množine T_2 a P_{T_4} je počet správne predikovaných hodnôt na T_4. Predikovaná hodnota je 0 ak $L(x) < 0.5$, 1 v opačnom prípade.

	T_1	T_2	T_1	T_2
k_0	-4.3115919	-5.0882126	N/A	N/A
k_1	0.8866516	1.9323949	3.4954483	7.5651639
k_2	0.0591388	-0.7750939	0.1915896	-2.5025246
k_3	-0.0639814	-0.9803391	-0.2694262	-4.1143708
k_4	1.2897811	1.6677921	1.2058379	1.3696831
k_5	-0.7649907	-0.1595905	-2.0485789	-0.4249205
k_6	-3.9572757	-6.5331655	-0.0547322	-0.0576496
k_7	-2.6652149	-3.1776481	-0.0374559	-0.0283028
k_8	-8.0943705	-2.8411723	-3.8456168	-1.1311728
k_9	1.4660135	2.4051818	1.9403395	4.4901156
k_{10}	-2.4364825	-0.9980142	-0.5607561	-0.1965638
k_{11}	-0.6484482	-3.5734406	-0.5706332	-2.7653036
k_{12}	-1.1370464	2.4089760	-1.2882769	3.4153358
k_{13}	0.5932194	0.0970350	0.0837849	0.0147686
k_{14}	4.1288503	-3.0481882	4.0063868	-2.9671991
k_{15}	-0.6396890	0.2338342	-0.5845121	0.2161926
k_{16}	0.5184831	0.4347640	0.4918839	0.4153258

Tabuľka 5.4: Koeficienty natrénovaných modelov logistickej regresie. Prvý stĺpec zodpovedá voľným trénovacím dátam a druhý stĺpec striktným. Tretí stĺpec obsahuje hodnoty prvého stĺpca vynásobené strednou hodnotou parametra s_i v príslušných trénovacích dátach. Podobne, v štvrtom stĺpci sú hodnoty druhého stĺpca vynásobené strednou hodnotou parametra s_i v príslušných trénovacích dátach.

5.5 Skórovanie udalostí

V tejto podkapitole experimentálne porovnáme rôzne prístupy ku výberu jednej udalosti z množiny kandidátov. Budeme za zaujímať o pravdepodobnosť, že uhádneme striktne správnu udalosť. Striktné porovnávanie udalostí sme zvolili preto, lebo ďalej budeme chcieť tieto algoritmy použiť na rekonštrukcie celých histórií, v ktorých je často dôležité určiť aj správny smer udalosti.

Tentoraz porovnáme štyri algoritmy, resp. štyri spôsoby skórovania.

B_0 je implementácia skórovania z predošlej práce [Hoz14], teda udalosti skórujeme na základe skórovacej funkcie 4.3.

B_1 je implementácia skórovania z tejto práce pre 16 parametrov, pričom pri trénovaní sme uvažovali všetky správne udalosti za pozitívne dáta. B_1' je to isté ako B_1, len používame všetkých 160 parametrov. B_2 je implementácia skórovania z tejto práce pre 16 parametrov, pričom sme na trénovanie použili striktné trénovacie dáta. B_2' je to isté ako B_2, len používame všetkých 160 parametrov.

B_4 slúži na znázornenie, ako veľmi dôležité sú skórovacie funkcie pre rekonštrukciu. Tento algoritmus nepoužíva skórovanie ale algoritmom A_1 (viď podkapitola 5.3) vygeneruje jediného kandidáta, ktorého prehlási za vybranú udalosť.

Pre každý algoritmus B a každú históriu H nás zaujíma pravdepodobnosť $q_{B,H}$, že daný algoritmus z náhodnej množiny kandidátov vyberie správnu udalosť. Túto pravdepodobnosť môžeme odhadnúť priemerom zo sto náhodne navzorkovaných množín. Pre jednu konkrétnu množinu kandidátov K pravdepodobnosť $q_{B,H,K}$ spočítame ľahko podľa vzťahu 4.1. $q_{B,H,K}$ je podiel súčtu skóre správnych udalostí a súčtu skóre všetkých udalostí v K. Množiny kandidátov generujeme algoritmom algoritmom A_1, takže $q_{B,H} \leq p_{A_1,H}$.

Výsledné štatistiky môžeme vidieť v tabuľke 5.5 a na obrázku 5.2. Všimnúť si môžeme niekoľko vecí (okrem toho, že pridaných 144 parametrov nepomáha, ako sme už spomínali v predošlej sekcii).

Len s jednoduchou skórovacou funkciou na základe dĺžky udalosti a čerešňovitosti nemáme veľkú šancu uhádnuť právnu udalosť.

Keď nenatrénujeme algoritmus na rozoznávanie smeru, tiež nebude sa mu dariť uhádnuť striktne správnu udalosť.

Ďalej si môžeme všimnúť, že ak má história málo udalostí, alebo málo atómov, nemusí to nutne znamenať, že sa rekonštruuje ľahko.

Posledná, najdôležitejšia vec je porovnanie algoritmu B_2 s algoritmom B_0 Pri niektorých históriách sa darí lepšie jednému algoritmu a v niektorých druhému. Dalo by sa povedať, že staršia skórovacia funkcia, použitá v algoritme B_0 je oveľa odvážnejšia vo svojich rozhodnutiach. Keď sa rozhodne, že udalosť je správna, dá jej veľmi vysoké

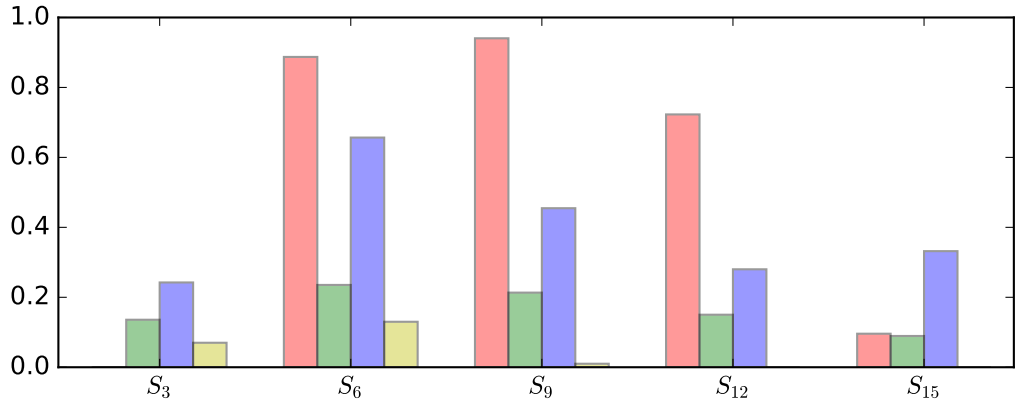
skóre. Keď sa však pomýli má to výraznejšie následky, ako napríklad pri rekonštrukcii poslednej udalosti zdanlivo jednoduchej histórie S_3. Algoritmus B_2 je umiernenejší, žiadnym rozhodnutím si nie je extrémne istý, ale zasa vždy má aspoň malú šancu, že udalosť uhádne.

Zatiaľ máme pocit, že algoritmus B_2 je lepší, najmä preto, že sa nám umiernenejší prístup zdá lepší na všeobecnú rekonštrukciu. Víťazný algoritmus však určíme až v ďalšej podkapitole.

Zároveň je obdivuhodné, že algoritmus B_0 bol taký úspešný ako bol, keďže jeho skórovacia funkcia bola určená empiricky zo šiestich parametrov. Nedosahoval by však také dobré výsledky, keby nebolo čerešňovitosti. V ďalšej kapitole otestujeme aj či si algoritmus poradí s históriami, keby dostával kandidátov vygenerovaných algoritmom A_0, ako to bolo v pôvodnej práci.

Algoritmus	q_{avg}	q_{10}	q_{50}	q_{90}
B_0	0.718	0.049	0.887	1.000
B_1	0.221	0.105	0.215	0.331
B'_1	0.219	0.094	0.214	0.287
B_2	0.540	0.248	0.546	0.757
B'_2	0.562	0.242	0.530	0.740
B_3	0.068	0.000	0.050	0.140

Tabuľka 5.5: Úspešnosť algoritmov na skórovanie udalostí. q_{avg}, q_{10}, q_{50}, q_{90} sú postupne priemer, 10. percentil, medián a 90. percentil z pravdepodobností $q_{B,H}$ pre simulované histórie.



Obr. 5.2: Pravdepodobnosť úspechu algoritmov B_0 (červený stĺpec), B_1 (zelený stĺpec), B_2 (modrý stĺpec) a B_3 (žltý stĺpec) pri vyberaní správnej udalosti z množiny kandidátov vo vybraných súčasných sekvenciách.

5.6 Rekonštrukcia histórií

Poďme preskúmať, ako si algoritmom darilo zrekonštruovať celé histórie. Zobrali sme algoritmy B_0, B_1, B_2, B_3 z predošlej podkapitoly a pre porovnanie sme pridali algoritmus B_4, ktorý používa skórovaciu funkciu z našej predchádzajúcej práce (podobne ako B_0) a pri generovaní kandidátov nepoužíva čerešňovitosť, tak ako sme ju nepoužívali ani v predchádzajúcej práci[Hoz14].

Každý algoritmus mal na zrekonštruovanie správnej histórie 1000 pokusov pričom hodnotiť ich len podľa toho, či sa im históriu podarilo alebo nepodarilo zrekonštruovať by nebolo dosť výpovedné. Úspešná rekonštrukcia celej histórie často závisí od šťastia. Namiesto toho sme rozlišovali štyri prípady. Buď sa algoritmu podarilo zrekonštruovať históriu aspoň desaťkrát z 1 000 pokusov, alebo sa algoritmu podarilo zrekonštruovať históriu aspoň raz ale menej ako desaťkrát, alebo sa algoritmu nepodarilo zrekonštruovať históriu, ale poradilo sa mu ju zrekonštruovať aspoň raz takmer celú (bez jednej udalosti) alebo sa mu ju vôbec nedarilo zrekonštruovať.

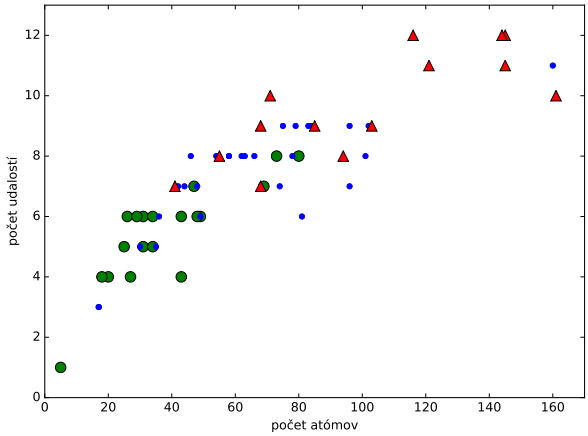
Zo sto simulovaných histórii sme odstránili tie, ktoré nebolo možné kompletne zrekonštruovať, kvôli nenájditeľnej delécii. Vo zvyšných 71 históriách pre každý algoritmus spočítame, koľkokrát nastal ktorý z prípadov, pričom tieto počty označíme postupne r_1, r_2, r_3, r_4. V tabuľke 5.6, vidíme výsledky experimentu.

Tentoraz je algoritmus B_2 jasný víťaz. Podarilo sa mu celkovo zrekonštruovať 54 zo 71 histórií. V tabuľke môžeme vidieť, že čerešňovitosť aj použitie logistickej regresie skutočne pomohlo pri rekonštrukcii.

Algoritmus	r_1	r_2	r_3	r_4
B_0	33	11	2	25
B_1	15	8	4	44
B_2	36	18	1	16
B_3	4	6	2	60
B_4	24	5	2	41

Tabuľka 5.6: Úspešnosť algoritmov pri rekonštruovani histórií.

Na záver ešte uvedieme jeden graf. Na obrázku 5.3, ako závisí obtiažnosť histórií od počtu udalostí v histórii a počtu atómov v súčasnej sekvencii. Keby sme zvýšili počet iterácii algoritmu z 1000 na povedzme 10000, pravdepodobne by sa nám podarilo zrekonštruovať ďalšie histórie.



Obr. 5.3: Obtiažnosť histórií podľa počtu udalostí a počtu atómov v súčasnej sekvencie. Zelené väčšie guličky zodpovedajú históriám, ktoré zrekonštruovali aspoň tri z algoritmov $B_0 \ldots B_4$. Modré, menšie guličky zodpovedajú históriám, ktoré zrekonštruoval správne aspoň jeden algoritmus. Štrnásť červených trojuholníkov zodpovedá ťažkým históriám, ktoré žiaden z algoritmov nezrekonštruoval počas 1000 pokusov.

Záver

Myslíme si, že rozdeliť problém rekonštrukcie na dve časti bolo dobré rozhodnutie, pretože sme tak mohli dôkladne preskúmať, ako úspešné dokážu byť algoritmy pri rekonštrukcii. Bez týchto poznatkov by sme ani nevedeli, či má zmysel hľadať medzi vygenerovanými históriami tú správnu.

V tejto práci sme tiež vylepšili obe dôležité časti algoritmu na rekonštrukciu histórie. Jednak sme využili aposteriórne pravdepodobnosti evolučných stromov na vypočítanie čerešňovitostí pre dvojice atómov. V druhom rade sme popísali udalosť pomocou 16 resp. 160 parametrov a naučili sme algoritmus celkom dobre rozoznávať správne a nesprávne udalosti.

Obe vylepšenia významne pomohli pri rekonštrukcii a umožnili nám posunúť hranicu histórií, ktoré sme schopní efektívne rekonštruovať. Mnohé reálne biologické dáta sa počtom atómov nachádzajú pod touto hranicou, takže už by sme mali byť schopní ich úspešne rekonštruovať.

V predošlej práci [Hoz14] sme navrhovali, že váhovaná kombinácia počtu udalostí a celkovej vierohodnosti by mohla byť dobré kritérium. Zatiaľ sme to však neskúmali.

Čo ďalej?

V práci sa dá pokračovať viacerými smermi. Na jednej strane sa pristúpiť k riešeniu druhej časti problému, ako spomedzi vygenerovaných udalostí vybrať tú správnu.

Keď sme ukázali, že dokážeme naučiť veľmi jednoduchý model, ktorým je logistická regresia na rozoznávanie správnych udalostí, dajú sa vyskúšať zložitejšie modely ako napríklad neurónové siete alebo rozhodovacie stromy. Možno tým algoritmus ešte viac zlepšíme.

Napokon sa dá rozšíriť algoritmus pre rekonštrukciu DNA sekvencií viacerých organizmov súčasne.

Literatúra

[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.

[HAS70] W. K. HASTINGS. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[Hoz14] Ján Hozza. Rekonštrukcia duplikačných histórií pomocou pravdepodobnostného modelu. Bachelor thesis, Comenius University in Bratislava, 2014. Supervised by Tomáš Vinař.

[HR03] JP Huelsenback and F. Ronquist. Mrbayes 3: Bayesian phylogenetic inference under mixed models. *BMC Bioinformatics*, 2003.

[JC69] T. H. Jukes and C. R. Cantor. *Evolution of Protein Molecules*. Academy Press, 1969.

[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.

[WD67] S. H. Walker and D. B. Duncan. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1):167–179, June 1967.