Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Rekonštrukcia evolučných histórií s duplikáciami pomocou simulovaného žíhania

Diplomová práca

2018
Matej Krajčovič

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Rekonštrukcia evolučných histórií s duplikáciami pomocou simulovaného žíhania

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	doc. Mgr. Bronislava Brejová, PhD.

Bratislava, 2018
Matej Krajčovič

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Matej Krajčovič
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 evolučných histórií s duplikáciami pomocou simulovaného žíhania
Reconstruction of Evolutionary Histories with Duplications Using Simulated Annealing

Anotácia: V genómoch sa vyskytujú oblasti, v ktorých počas evolučnej histórie došlo k viacnásobnej duplikácii časti sekvencie. Cieľom práce je vyvinúť nástroj na rekonštrukciu evolučnej histórie takejto oblasti pre vstupnú sekvenciu DNA. Nakoľko problémy rekonštrukcie histórie v zložitejších evolučných modeloch sú typicky NP ťažké, nástroj bude založený simulovanom žíhaní, čo je jedna zo zaužívaných metaheuristík na riešenie ťažkých problémov.

Vedúci: doc. Mgr. Bronislava Brejová, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 11.10.2016

Dátum schválenia: 14.12.2016 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Abstrakt

V dôsledku rozsiahlych duplikácií v pomerne krátkej časti DNA sekvencie vznikajú génové zhluky so zložitou štruktúrou. Keďže sa tieto zhluky nachádzajú aj v ľudskom genóme a nachádzajú sa v nich gény, ktoré majú súvis s viacerými chorobami ako napríklad rakovina prostaty, chceme zistiť, akým spôsobom vznikli. Tento problém sa nazýva rekonštrukcia histórie.

V tejto práci zlepšujeme výsledky predošlého algoritmu pomocou metódy simulovaného žíhania. Keďže má táto metóda viacero parametrov, skúšame rôzne prístupy k riešeniu a následne ich porovnávame. Tiež predstavujeme nový spôsob porovnávania rekonštrukcií.

Kľúčové slová: DNA sekvencia, génový zhluk, rekonštrukcia, simulované žíhanie, evolúcia

Abstract

Gene clusters with a complex structure are a result of long duplications in relatively short part of DNA sequence. Because there are such clusters in the human genome and contain genes linked with various diseases, e.g. prostate cancer, we want to find out the way they were created. This problem is called history reconstruction.

In this thesis, we improve results of the previous algorithm with use of simulated annealing. Because this method has a lot of parameters, we try various approaches and compare them. We also present a new method of comparing reconstructions.

Keywords: DNA sequence, gene cluster, reconstruction, simulated annealing, evolution

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V bunkách živých organizmov sa nachádza DNA, v ktorej sú zakódované ich genetické informácie. Pri prenose tejto informácie z predkov na potomkov sa občas vyskytujú zmeny, medzi ktoré patria aj duplikácie a delécie, ktoré kopírujú alebo vymazávajú dlhé úseky DNA. Vplyvom týchto udalostí na pomerne krátkych úsekoch vznikajú génové zhluky so zložitou štruktúrou. Našou úlohou je pre zadanú DNA sekvenciu zistiť jej históriu.

Tomuto problému sa v minulosti venovalo viacero prác, ktorým sa podarilo vyriešiť tento problém pre jednoduchšie prípady. V tejto práci budujeme na algoritmoch, ktoré sa vyvynuli v minulosti, a pomocou metódy simulovaného žíhania sa snažíme zlepšiť ich výsledky. Tiež navrhujeme alternatívny spôsob porovnávania výsledkov algoritmov, ktorý nezisťuje iba rovnosť histórií, ale viem určiť ich podobnosť.

V prvej kapitole predstavujeme problém, ktorý riešieme a definujeme viacero pojmov.

V druhej kapitole popisujeme dôležité časti z dvoch prác, ktorých kód priamo používame.

V tretej kapitole sa venujeme simulovanému žíhaniu. Najskôr popisujeme teóriu, na ktorej je založené, následne ukážkové riešenie problému obchodného cestujúceho a vo zvyšku kapituly popisujeme, ako simulované žíhanie používame na riešenie problému rekonštrukcie histórie.

Vo štvrtej kapitole popisujeme náš nový spôsob porovnávania histórií.

V piatej kapitole porovnávame výsledky rôznych metód, ktoré spomíname v predošlých kapitolách, a tiež porovnanie výsledných algoritmov.

V poslednej šiestej kapitole popisujeme problémy a výzvy pri implementácie práce.

Zdrojový kód algoritmu je možné nájsť na stránke `https://github.com/matejkrajcovic/hroch`.

Kapitola 1

Predstavenie problému

V tejto kapitole definujeme problém, ktorý riešime, a vysvetlíme pojmy, ktoré budeme používať v celej práci.

1.1 DNA sekvencia a udalosti

V bunkách každého živého organizmu sa nachádza DNA, v ktorej je zakódovaná jeho genetická informácia. Skladá sa z *báz* - adenín, cytozín, tymín a guanín - ktoré budeme ďalej označovať ich prvými písmenami: A, C, T, G. DNA má formu dvojzávitnice, ktorej dve vlákna majú opačné smerovanie a oproti sebe sa nachádzajú komplementárne bázy - A oproti T, C oproti G a naopak. V súčasnosti máme k dispozícií technológie, ktoré umožňujú osekvenovať DNA jedinca a získať konečnú postupnosť písmen z abecedy {A, C, T, G}, ktoré prislúchajú jednotlivým bázam.

Počas prenosu genetickej informácie z predka na potomkov môžu pri kopírovaní DNA nastať *mutácie*, ktoré v nej spôsobia malé, ale aj rozsiahle zmeny. V práci využijeme určitým spôsobom aj lokálne zmeny, ale predovšetkým nás budú zaujímajú tie rozsiahle, ktoré budeme nazývať *udalosti*. Patria medzi ne *duplikácia*, *duplikácia s inverziou* a *delécia*.

- *Duplikácia* skopíruje dlhú časť sekvencie zo zdrojového miesta a vloží ju na cieľové miesto v sekvencii, ktoré ale nemôže byť vnútri v intervale, z ktorého kopírujeme.
- *Duplikácia s inverziu* podobne ako duplikácia skopíruje dlhú časť sekvencie zo zdrojového miesta, ale pred vložením na cieľové miesto v sekvencii zmení bázy na komplementárne a kopírovaný reťazec otočí. Z biologického pohľadu to zodpovedá skopírovaniu časti sekvencie z druhého vlákna.
- *Delécia* odstráni dlhú časť sekvencie.

1.2 Duplikačná história

Postupnosť udalostí budeme nazývať *duplikačná história*, skrátene *história*. Pôvodnú sekvenciu, z ktorej história začala, nazveme *ancestrálna sekvencia*, a sekvenciu, ktorou história končí, nazveme *súčasná sekvencia*.

Každá udalosť má určený čas, kedy sa vyskytla a ktorý meriame v zaužívaných jednotkách, kde hodnote 1 zodpovedá čas, za ktorý očakávame, že na DNA sekvencii dĺžky l sa vyskytne l lokálnych mutácii. Rozdiel časov medzi prvou a poslednou udalosťou v histórii nazveme *dĺžkou histórie*. V práci budeme pracovať predovšetkým s históriami, ktorých dĺžka je 0.04 alebo 0.06, čo zodpovedá 25 až 33 miliónov rokov, respektívne 37 až 49.5 miliónov rokov [4].

Príklad duplikačnej histórie je možné vidieť na obrázku 1.2. Najskôr boli duplikované atómy “2 3” a následne “3 4 2”, z ktorých ale bol vymazaný atóm “4”.

1.3 Atómy a stromy atómov

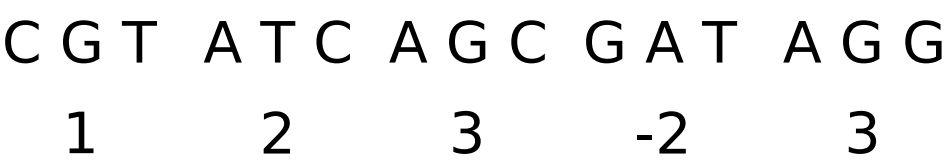
DNA sekvencie, s ktorými budeme pracovať, sú veľmi dlhé a manipulovať s nimi by bolo časovo aj pamäťovo náročné a chceme ich preto skrátiť a zjednodušiť. Keďže uvažujeme udalosti, ktoré kopírujú dlhé úseky DNA, môžeme predpokladať, že súčasná sekvencia DNA bude obsahovať veľa veľmi podobných úsekov. Tieto úseky nazveme *atómy*. Atómy majú rovnaký *typ*, ak sa navzájom veľmi podobajú a zároveň sú veľmi odlišné od atómov iných typov. Každý typ má pridelené jednoznačné celé kladné číslo, ktoré ho symbolizuje. Sekvenciu DNA vieme následne prepísať na postupnosť atómov, ktorou bude postupnosť čísel typov atómov, pričom záporné čísla budú symbolizovať inverziu atómu.

Dôvod, prečo nepožadujeme, aby boli atómy v rámci jedného typu rovnaké, ale iba podobné, je kvôli lokálnym mutáciam, ktoré spôsobujú mierne odlišnosti medzi atómami, a tiež kvôli chybám sekvenovania. Ak zavedieme minimálnu dĺžku atómov, môžeme predpokladať, že všetky atómy rovnakého typu naozaj vznikli z jedného atómu v ancestrálnej sekvencii pomocou duplikácií, lebo je nepravdepodobné, že by dlhé vysoko podobné sekvencie vznikli samovoľne pomocou lokálnych mutácií.

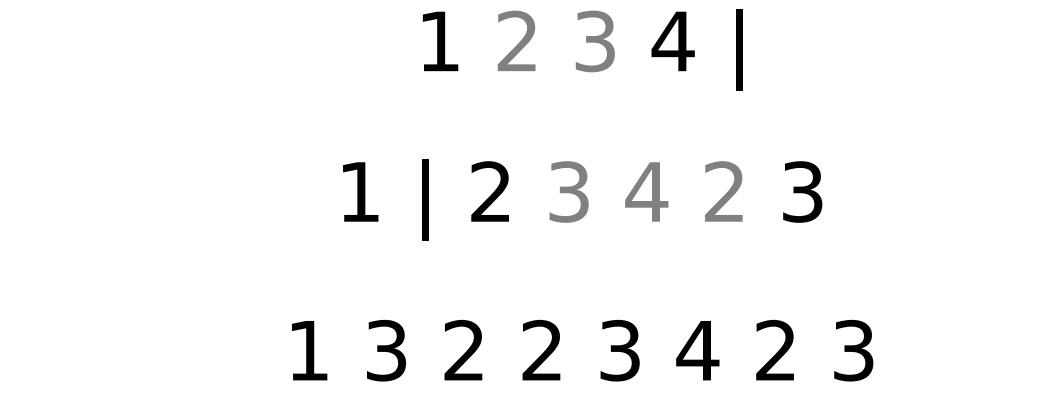
Problému určovania atómov sa nebudeme v práci venovať, ale použijeme skript, ktorý vie pre DNA sekvenciu zistiť sekvenciu atómov. Pomocou atomizácie vieme skrátiť dĺžku súčasnej sekvencie z 200 000 až 400 000 báz na 20 až 200 atómov.

Ukážku atomizácie je možné vidieť na obrázku 1.1.

Na základe sekvencií atómov súčasnej sekvencie je možné z atómov rovnakého typu vytvoriť strom atómov. Jeho listami sú atómy prítomné v súčasnej postupnosti atómov a vnútornými vrcholmi sú atómy, ktoré boli prítomné v minulých sekvenciách atómov. Pre duplikačnú históriu znázornenú na obrázku 1.2 sú zodpovedajúce stromy atómov



Obr. 1.1: Atomizácia DNA sekvencie

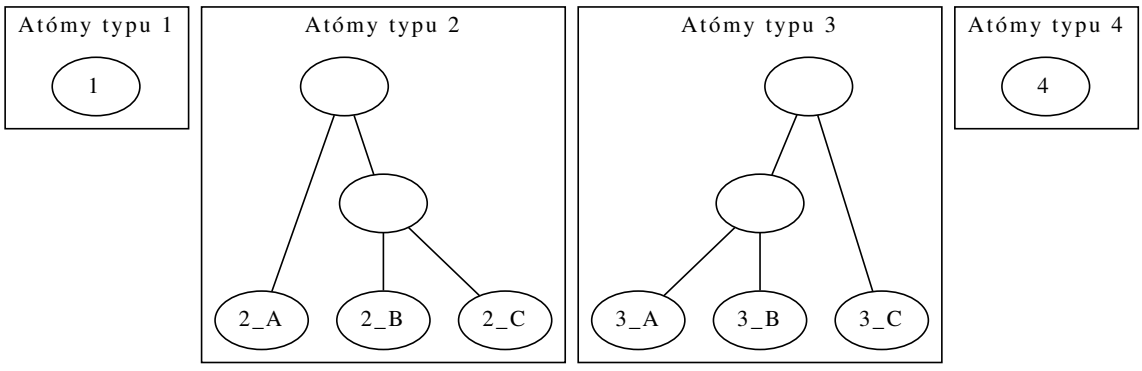


Obr. 1.2: Možná duplikačná história - v dolnom riadku je postupnosť atómov súčasnej sekvencie DNA a vo vrchnom pôvodná. Šedé časti sekvencií sú duplikované a vložené na miesta označené zvyslou čiarou

znázornené na obrázku 1.3.

1.4 Význam rekonštrukcie génových zhlukov

Génové zhluky tvoria približne 5% ľudského genómu a nachádzajú sa v nich gény, ktoré súvisia s rakovinou prostaty, Alzheimerovou chorobou a inými chorobami [7]. Ak by sa nám podarilo pre tieto úseky ľudského genómu zistiť ich duplikačné histórie, lepšie by sme porozumeli, kedy pribúdali vplyvom duplikácií rôzne gény.



Obr. 1.3: Stromy atómov zodpovedajúce v histórií zobrazenej na obrázku 1.2

Kapitola 2

Existujúce práce

V tejto kapitole popíšeme dve práce, na ktoré úzko súvisia s našou prácou. Prvou je "Rekonštrukcia histórií génových zhlukov" [4], na ktorú priamo nadväzujeme a upravujeme jej zdrojový kód, a druhou je "Časovanie udalostí pri inferencii duplikačných histórií" [1], ktorú používame na výpočty. V oboch prípadoch nepopíšeme všetky detaily, ale skôr chceme vysvetliť princípy, na ktorých sú založené, a na ktoré sa budeme neskôr odvolávať.

2.1 Rekonštrukcia histórií génových zhlukov

Prvá práca, ktorej sa budeme venovať je diplomová práca Mgr. Jána Hozzu. Rieši rovnakú úlohu ako my - pre zadanú súčasnú sekvenciu atómov a stromy atómov sa snaží zrekonštruovať správnu históriu. Kostra jeho algoritmu rekonštrukcie histórie je nasledovná:

1. Opakuj, pokiaľ sa v sekvencii nenachádza každý typ atómu práve raz:
   (a) navzorkuj si možné udalosti
   (b) oskóruj ich
   (c) náhodne vyber udalosť pri zohľadnení skóre
   (d) odstráň na sekvencii následky udalosti a získaj novú sekvenciu

V každej iterácii sa odstráni aspoň jedna kópia niektorého atómu a cyklus sa vždy zastaví po konečnom počte iterácií.

2.1.1 Vzorkovanie udalostí

V práci [6] je popísané, že v prípade sekvencie atómov dĺžky n je možných $\Theta(n^3)$ udalostí, ak uvažujeme iba duplikácie bez delécií, $\Theta(n^4)$ v prípade jednej delécie dĺžky

1 a $\Theta(n^5)$ v prípade jednej delécie ľubovoľnej dĺžky. To znemožňuje prehľadávať všetky možné udalosti a potrebujeme mať spôsob, ako si efektívne navzorkovať dobré udalosti. Je ponúknuté nasledovné riešenie.

Pre udalosť u vieme vypočítať jej skóre pomocou funkcie $s(u)$:

$$s(u) = p_{dup}^{l} * p_{del}^{k} * q_{del}^{d_1+\cdots+d_k}$$

kde l je dĺžka duplikácie, k je počet delécií a d_i sú dĺžky delécií. p_{dup}, p_{del} a q_{del} sú parametre nastavené na hodnoty $p_{dup} = 2$, $p_{del} = 0.05$ a $q_{del} = 0.5$.

Pre takto definovanú funkciu existuje efektívny algoritmus, ktorý vráti udalosť u s pravdepodobnosťou

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

kde U je množina všetkých možných udalostí.

Ak by sme uvažovali iba duplikácie bez inverzie a delécií, vieme vytvoriť orientovaný ohodnotený graf, ktorého vrcholy sú dvojice (i, j) pre $i \neq j$, ak sa na pozíciach i a j nachádzajú atómy rovnakého typu a rovnakej orientácie. Medzi vrcholmi (i, j) a $(i + 1, j + 1)$ sa vytvoria hrany ceny 2. Nakoniec sa pridajú špeciálne vrcholy B a E. Z vrcholu B bude viesť cesta ceny 2 do každého vrcholu okrem vrcholu E a analogicky z každého vrcholu okrem vrcholu B bude viesť cesta ceny 1 do vrcholu E. Cena cesty je súčin cien hrán na ceste. Do každého vrcholu sa uloží súčet ciest, ktoré vedú z vrcholu B doň.

Na obrázku 2.1 je vidieť graf, ktorý dostaneme pre postupnosť “1 2 -1 2 1 2 -1”. Všimnime si, že cena každej cesty zodpovedá skóre, ktoré by získala pomocou funkcie $s(u)$.

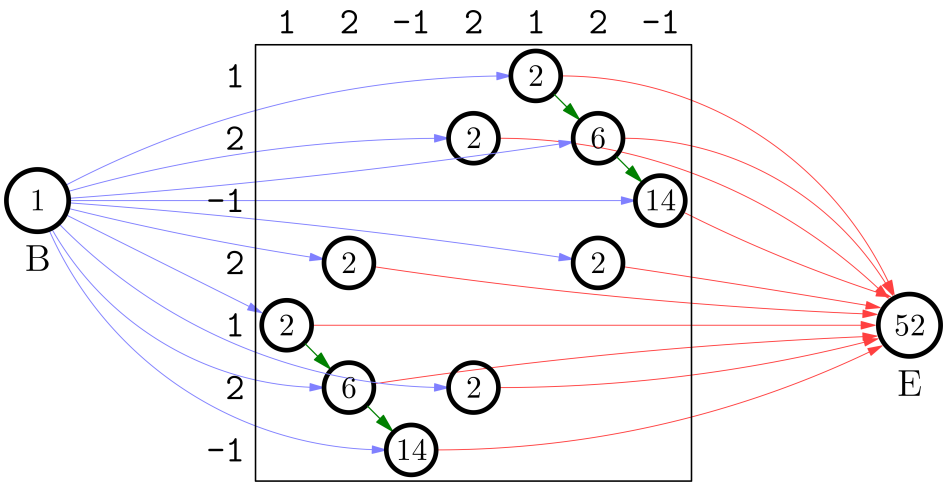
Keď chceme z tohto grafu vzorkovať udalosti, stačí ho prechádzať z vrcholu E opačným smerom po hranách. Vždy si náhodne vyberieme niektorú z možných hrán podľa pomerov ich hodnôt v nich.

V pôvodnej práci je popísané, ako je možné graf tohto typu rozšíriť na inverzie a delécie.

Časová zložitosť navzorkovania k udalostí je $O(n^2+nk)$. Aby celková zložitosť nebola viac ako $O(n^2)$, vygeneruje sa $O(n)$ udalostí.

2.1.2 Skórovanie udalostí

Po navzorkovaní udalostí je potrebné ich ešte oskórovať sofistikovanejším spôsobom, ktorý bude hodnotiť viac parametrov, ako iba dĺžku duplikácie, počet delécií a súčet dĺžok delécií. Pri skórovaní sa používa 16 základných parametrov a 144 odvodených, ktoré vznikli umocnením na e, vynásobením alebo vydelením základných parametrov.



Obr. 2.1: Graf pre postupnosť "1 2 -1 2 1 2 -1". Obrázok pochádza z práce [4].

Nebudeme spomínať všetky základné parametre, ale iba pár z nich, aby sme demonštrovali, že sa jedná o parametre udalosti, ktoré by intuitívne mohli vypovedať o pravdepodobnosti jej správnosti.

Označme si S_a sekvenciu pred vykonaním udalosti a S_b sekvenciu po vykonaní. Medzi niektoré základné parametre patrili:

- logaritmus dĺžky S_b,
- logaritmus počtu rôznych typov atómov v S_b,
- logaritmus dĺžky duplikácie,
- $log(1 + v)$, kde v je vzdialenosť duplikácie,
- počet duplikácií,
- logaritmus počtu atómov v duplikáciach.

Viaceré parametre sú logaritmy a nie pôvodné hodnoty, aby neboli hodnoty niektorých parametrov oveľa väčšie ako iných.

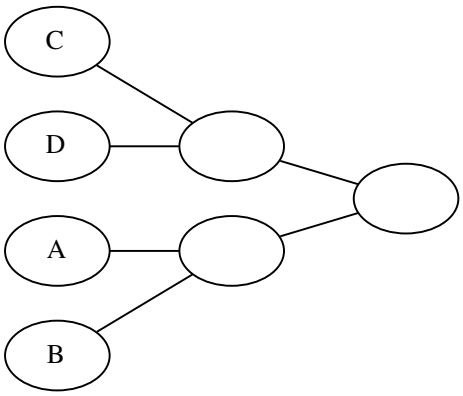
Ak k parametrom poznáme koeficienty k_i, vieme získať skóre vzťahom

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

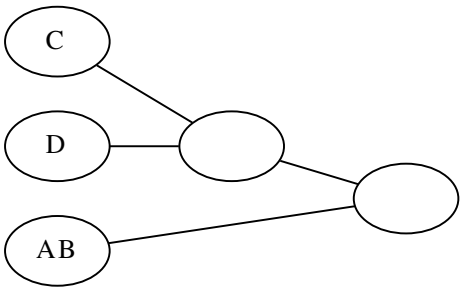
kde funkcia $L(x)$ sa nazýva sigmoida, je nelineárna a rastúca, mapuje reálne čísla do rozsahu $(0, 1)$ a je zadaná vzťahom

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

Hodnoty koeficientov k_i sa zistia logistickou regresiou, ktorá sa natrénuje na testovacích dátach.



Obr. 2.2: Strom atómov pred zlúčením čerešne



Obr. 2.3: Strom atómov po zlúčení čerešne

2.1.3 Čerešňovitosť

Zaujímavá vlastnosť, ktorá sa používa pri skórovaní udalostí je čerešňovitosť stromov atómov. Pripomeňme, že pre každý typ atómov máme k dispozícii nezakorenený strom atómov daného typu, v ktorom sú blízko pri sebe navzájom podobné atómy v zmysle počtu substitúcií medzi zodpovedajúcimi sekvenciami. Dva atómy tvoria "čerešňu", ak majú spoločného suseda. Vizuálne pripomínajú dve čerešne a ich sused je stopka.

Na obrázku 2.2 je vidieť strom atómov pre atómy A, B, C a D. Dvojice atómov A, B a C, D tvoria čerešne, lebo majú spoločných susedov. V prípade, že jeden z atómov A, B sa nachádzal v sekvencii, ktorá bola duplikáciou kopírovaná, a druhý atóm sa nachádzal v sekvencii, ktorá bola vložená, je nutné po tejto udalosti zľúčiť oba atómy do jedného v strome atómov. Čerešna sa nahradí vrcholom AB, ktorý reprezentuje predka oboch atómov. Výsledný strom atómov je vidieť na obrázku 2.3.

Stromy sú uložené takým spôsobom, že je možné overiť v konštantnom čase, či dva atómy tvoria čerešňu. Pri skórovaní udalosti sú použité parametre založené na počte použitých čerešní.

Tabuľka 2.1: Parametre pravdepodobnostného modelu použitého na generovanie simulovaných histórií a počítanie vierohodností rekonštrukcií

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

2.1.4 Generovanie simulovaných histórií

Dôležitou súčasťou práce je tiež generátor simulovaných histórií. Na vstupe dostane dĺžku ancestrálnej sekvencie DNA, ktorú vygeneruje, a čas, aký sa má vyvýjať. Reálnym dátam zodpovedá dĺžka 100 000 báz a čas 0.04 našich jednotiek času, ale je možné ich upraviť a dostať jednoduchšie alebo zložitejšie histórie.

Časy udalosti sa modelujú Poissonovým rozdelením s parametrom λ, ktorý zodpovedá počtu udalostí za jednu jednotku času. Pokiaľ je ešte k dispozícii čas, zvolíme čas najbližšej udalosti. Dovtedy simulujeme substitúcie na bázach pomocou Jukes-Cantorovho modelu. S pravdepodobnosťou p_{del} je udalosť delécia, s pravdepodobnosťou p_{inv} je inverzia a s pravdepodobnosťou $1 - p_{del} - p_{inv}$ je bežná duplikácia bez inverzie a delécií. S pravdepodobnosťou 0.5 je orientácia udalosti doľava a s pravdepodobnosťou 0.5 doprava. Dĺžka duplikovanej časti je modelovaná geometrickým rozdelením so strednou hodnotou m_l a vzdialenosť zdroja a cieľa duplikácie je modelovaná geometrickým rozdelením so strednou hodnotou m_v - ak by sa pri oboch hodnotách presiahli okraje sekvencie, vygenerujú sa nové hodnoty. V tabuľke 2.1 sú hodnoty spomenutých konštánt.

Spomenutý Jukes-Cantorov model hovorí, že pravdepodobnosť mutácie bázy x na bázu y za čas t je daná vzťahom

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

čiže nezávisí na pôvodnej a novej hodnote bázy.

2.2 Časovanie udalostí pri inferencii duplikačných histórií

Druhá práca, ktorej sa budeme venovať je bakalárska práca Mgr. Michala Anderleho, ktorá rieši odlišný problém. Väčšina algoritmov na rekonštrukciu histórií sa sústreďuje na určenie správnych udalostí a nezaoberá sa určovaním časov medzi udalosťami, čo má tiež nezanedbateľný vplyv na kvalitu rekonštrukcie. Ani predošlá práca nezisťuje časy

medzi udalosťami, ale čas medzi udalosťami nastavuje na konštantnú hodnotu 0.01. Úlohou, ktorú táto práca riešila, je dostať na vstupe rekonštrukciu histórie spolu so zarovnaniami atómov a postupne upravovať časy udalostí tak, aby sa nezmenilo poradie udalostí a ani celkový čas histórie, ale aby mala výsledná história lepšie nastavené časy udalostí.

V práci je definovaný pravdepodobnostný model evolúcie zodpovedúci modelu použitému v časti 2.1.4 o generovaní simulovaných histórií a výpočet, ktorým sa zistí pravdepodobnosť, že daný model vygeneruje práve túto históriu. Samotný výpočet pravdepodobnosti tu nebudeme uvádzať, ale spomenieme aspoň, že výsledná pravdepodobnosť závisí od typov udalostí, časov, kedy sa udiali, dĺžky duplikovanej sekvencie, vzdialenosti medzi zdrojovou a cieľovou pozíciou a tiež od množstiev substitúcií, ktoré nastali v atómoch. Po zarátaní všetkých týchto faktorov je výsledná pravdepodobnosť príliš malá, aby mohla byť uložená v bežných dátových typoch na prácu s reálnymi číslami, a preto je jej hodnota po celý čas uložená vo forme logaritmu a namiesto násobenia sa jednotlivé časti pravdepodobnosti sčítavajú.

Pri procese časovania udalostí sa 100krát zopakuje operácia, pri ktorom sa náhodne vyberie udalosť a upraví jej čas tak, aby sa maximalizovalo zlepšenie pravdepodobnosti celej histórie.

Kapitola 3

Simulované žíhanie

V tejto kapitole definujeme simulované žihanie a popisujeme, ako ho používame v práci.

3.1 Úvod

Simulované žíhanie je pravdepodobnostná optimalizačná metóda na hľadanie približnej hodnoty globálneho extrému zadanej funkcie. Často sa úspešne používa na riešenie problémov, ktoré sa vyznačujú veľkým stavovým priestorom a množstvom lokálnych extrémov.

Simulované žíhanie ako metódu nezávisle na sebe publikovali v roku 1983 Kirkpatrick, Gelett a Vecchi [5] a Černý v roku 1985 [3].

3.2 Pôvod

Metóda simulovaného žíhania je inšpirovaná fyzikálnym procesom pri spracovaní kovu - metalurgii. Žíhanie označuje proces, pri ktorom sa tuhé teleso zahreje na vysokú teplotu a postupným ochladzovaním sa odstraňujú defekty a zvyšuje pevnosť materiálu.

Pre každý kov existuje také usporiadanie atómov do kryštalickej štruktúry, pri ktorom je minimalizovaná vnútorná energia, existuje málo defektov spôsobených zle umiestnenými atómami a v dôsledku toho je kov pevný. Pri procese žíhania sa najskôr kov pri teplote dostatočne vysokej na rozbitie existujúcich kryštálov roztopí a častice sa voľne a náhodne pohybujú. Pri vysokej teplote sa často dostanú aj do zdanlivo horších pozícií, ktoré spôsobia zvýšenie vnútornej energie, ale sú potrebné, lebo vďaka nim sa možno vedia následne dostať do ešte lepších pozícií. So znižujúcou teplotou sa častice pohybujú pomalšie a náhle ochladenie by spôsobilo, že častice "zamrznú" na aktuálnych pozíciach, čo nemusí byť optimálne. Ak ale teplotu znižujeme dostatočne pomaly, aj rýchlosť častíc sa znižuje pomaly a tiež sa znižuje pravdepodobnosť, že sa dostanú do horších pozícií. Nastáva teda čoraz lokálnejšie optimalizovanie pozícií častíc

- najskôr sú testované rôzne pozície, niektoré aj horšie, ale postupne sa začínajú stabilizovať. Počiatočná vysoká teplota je výhodná v tom, že umožňuje dostať sa z lokálneho optima.

3.3 Popis metódy

Existuje viacero variácií simulovaného žíhania, ktoré ale zdieľajú základné princípy. Najskôr zadefinujeme niektoré pojmy a popíšeme pseudokód, ktorý sa používa v pôvodnom článku [3].

Definícia 1. *Stavový priestor je množina stavov, ktorú prehľadávame a hľadáme v nej riešenie.*

Definícia 2. *Susedia stavu x tvoria množinu stavov, ktoré sú v určitom zmysle podobné stavu x. Susedia sú väčšinou definovaní operáciou, ktorá mierne zmení pôvodný stav. Ak by boli susedia pôvodnému stavu až príliš podobní, dochádzalo by iba k lokálnej optimalizácii. Ak by naopak boli susedia od pôvodného stavu veľmi odlišní, nedochádzalo by postupnému zlepšovaniu stavu, ale priebeh by skôr pripomínal náhodné generovanie stavov.*

Definícia 3. *Skórovacia funkcia slúži na kvantifikovanie, nakoľko je stav dobré riešenie problému. Predpokladajme ďalej, že hľadáme globálne minimum a teda očakávame nižšie hodnoty funkcie pre lepšie stavy ako horšie.*

Definícia 4. *Teplota je dôležitý parameter metódy, ktorý ovplyvňuje pravdepodobnosť akceptovania horšieho stavu. Rozvrh chladnutia popisuje zmeny teploty - či sa mení po každom kroku alebo po viacerých krokoch, či sa zníži o konštantnú hodnotu alebo je zmena popísaná inou funkciou.*

Definícia 5. *Akceptačná pravdepodobnosť určuje pravdepodobnosť prechodu do nového stavu. Ak je skóre nového stavu lepšie ako skóre súčasného stavu, vždy sa akceptuje. Ak je skóre nového stavu horšie, akceptačná pravdepodobnosť je závislá od rozdielov skóre a aktuálnej teploty.*

Nasleduje pseudokód simulovaného žíhania:

Krok 1: Zvoľ náhodný stav a ulož ho do premennej s.

Krok 2: Vypočítaj skóre stavu s a ulož ho do premennej d.

Krok 3: Zvoľ náhodného suseda stavu s a ulož ho do premennej s'.

Krok 4: Vypočítaj skóre stavu s' a ulož ho do premennej d'.

Krok 5: Ak $d' < d$, presuň sa do kroku 8, ak nie, vygeneruj náhodné číslo z rozsahu $\langle 0, 1 \rangle$.

Krok 6: Ulož do premennej a hodnotu akceptačnej pravdepodobnosti pri teplote T a rozdielu skóre $d - d'$.

Krok 7: Ak $x < a$, presuň sa do kroku 8, ak nie, presuň sa do kroku 9.

Krok 8: (Akceptovanie nového stavu) Nastav $s = s'$ a $d = d'$. Vhodne uprav teplotu T podľa rozvrhu chladnutia. Presuň sa do kroku 3.

Krok 9: (Neakceptovanie nového stavu) Vhodne uprav teplotu T podľa rozvrhu chladnutia. Presuň sa do kroku 3.

Takto popísaný algoritmus sa nezastaví a je preto ešte potrebné určiť podmienku ukončenia. Napríklad je možné skončiť po konštantnom počte kôl alebo ak sa skóre už dlho nezlepšilo.

3.4 Problém obchodného cestujúceho

Učebnicový príklad aplikácie simulovaného žihania je problém obchodného cestujúceho, pri ktorom je zadaný zoznam miest a dĺžky ciest medzi nimi a hľadá sa najkratšia cesta, pri ktorej sa navštívia všetky mestá a cestujúci sa vráti do pôvodného mesta. Problém je NP-ťažký, ale v pôvodnom článku o simulovanom žíhaní sa podarilo nájsť riešenie veľmi blízko globálneho optima v čase výrazne kratšom ako overovanie všetkých $n!$ možností.

Stavom je permutácia miest a vyjadruje postupnosť, v akej budú navštívené, pričom z posledného mesta v zozname sa následne ešte cestujúci presunie do prvého miesta. Skórovacia funkcia je súčet dĺžok ciest medzi mestami, ktoré sa nachádzajú po sebe v zozname, a dodatočnej cesty späť do počiatočného mesta.

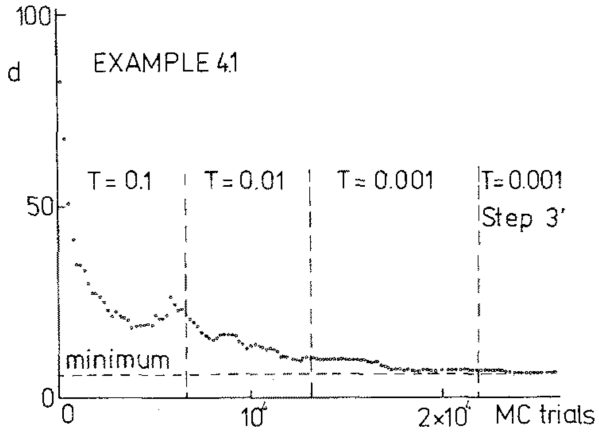
Nasledujúci stav sa generuje pomocou operácie, ktorá v stave vymení dve mestá na pozíciach i a j a otočí cestu medzi nimi. Premenná i sa na začiatku žíhania inicializuje na hodnotu 0 a zvyšuje sa o 1 po každom akceptovaní nového stavu a v prípade, že je väčšia ako počet miest, sa opäť nastaví na hodnotu 0. Druhá premenná j sa zvolí náhodne.

Akceptačná pravdepodobnosť je určená nasledovnou funkciou, kde d je skóre súčasného stavu, d' je skóre nového stavu a T je aktuálna teplota:

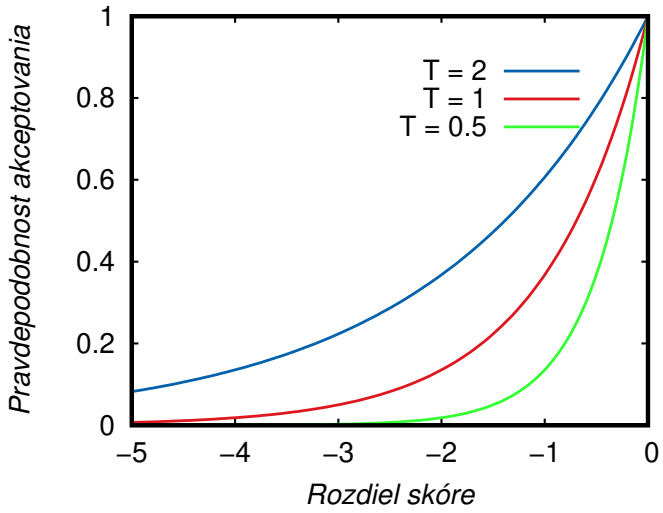
$$e^{(d-d')/T}$$

.

Pre tri rôzne hodnoty teploty ilustrujeme na grafe 3.2 priebeh tejto funkcie.



Obr. 3.1: Priebeh žíhania



Obr. 3.2: Pravdepodobnosť akceptovania v závislosti od rozdielu skóre pre rôzne hodnoty teplôt

Počiatočná teplota bola nastavená na hodnotu 0.1 a zmenšila sa na desatinu vždy po 6600 krokoch.

Celkovo po 25000 krokoch sa podarilo nájsť optimálne riešenie pre 100 miest, ktoré boli rovnomerne uložené na jednotkovej kružnici.

Postupný vývoj skóre je možné vidieť na obrázku 3.1.

Problém obchodného cestujúceho sa podarilo úspešne vyriešiť vďaka tomu, že autori dobre definovali generovanie susedov a mali rýchlu a presnú skórovaciu funkciu. V nasledujúcich podkapitolách predstavíme náš prístup k simulovanému žíhaniu na riešenie rekonštrukcií histórií.

3.5 Aplikácia simulovaného žíhania na rekonštrukciu histórií

Pre úspešné aplikovanie simulovaného žíhania na problém rekonštrukcie histórie musíme rovnako ako v prípade problému obchodného cestujúceho zodpovedať nasledujúce otázky:

1. Ako zakódovať riešenie problému do stavu?
2. Ako efektívne generovať susedov stavu?
3. Ako určiť skóre stavu?
4. Ako nastaviť počiatočnú teplotu a rozvrh chladnutia?
5. Kedy skončiť žíhanie?
6. Ako je definovaná akceptačná pravdepodobnosť?

Stavom bude priamo rekonštrukciu histórie. Skórovanie a generovanie susedov rekonštrukcií podrobne vysvetlíme v nasledujúcich častiach.

Akceptačnú pravdepodobnosť budeme počítať rovnakou funkciou ako v ukážke riešenia problému obchodného cestujúceho.

Počiatočnú teplotu nastavíme v prípade prvej skórovacej funkcie na 0.4 a v prípade druhej na 100. Rozdiel je spôsobený tým, že v prípade prvej je rozdiel skóre o 1 dôležitý, ale v prípade druhej skórovacej funkcie sú jej hodnoty z oveľa väčšieho rozsahu a chceme povoliť akceptovanie horších rekonštrukcií aj v prípade väčšieho rozdielu skóre.

Pred začatím simulovaného žíhania nezávisle vygenerujeme 10 rekonštrukcií, aby sme začali žíhanie s pomerne dobrou rekonštrukciou. Keď sme tento krok nepoužívali a začínali sme žíhanie náhodnou rekonštrukciou, dostávali sme o niečo horšie výsledky, lebo sme niekedy začali optimalizovaním pomerne zlej rekonštrukcie. Následne rozdelíme kroky žíhania na tretiny a pri prechode do ďalšej tretiny sme teplotu znížili na tretinu.

3.6 Skórovacia funkcia

3.6.1 Počet udalostí

Najjednoduchšia metóda skórovania rekonštrukcie a zároveň metóda, ktorá bola použitá aj v minulej práci, je založená na počte udalostí, ktoré potrebovala rekonštrukcia na vysvetlenie súčasnej sekvencie. Pripomeňme, že udalosti sú relatívne vzácne. Počas testovania rekonštrukcií sa nám občas stalo, že pri rekonštruovaní súčasnej sekvencie,

ktorá zodpovedala histórií s 20timi udalosťami, sme dostali rekonštrukcie, ktoré mali 50 alebo aj viac ako 100 udalostí. Je jednoduché vidieť, ako dokážeme pre ľubovoľnú rekonštrukciu zvýšiť počet jej udalostí - z ľubovoľnej duplikácie, ktorá kopíruje aspoň dva atómy, vieme vytvoriť viac duplikácií, ktoré kopírujú danú sekvenciu postupne po častiach.

Nevýhoda tejto metódy je v tom, že existuje veľa rekonštrukcií s rovnakým počtom udalostí a teda aj skóre a nevieme medzi nimi rozhodnúť, ktorá z nich je lepšia.

3.6.2 Vierohodnosť

Druhá metóda je založená na počítaní vierohodnosti pomocou kódu, ktorý bol výsledkom práce [1] a popisujeme ju v časti 2.2. Pripomeňme, že metóda z danej práce počítala pravdepodobnosť, že pravdepodobnostný model vygeneruje práve túto históriu.

Na rozdiel od zistenia počtu použitých udalostí je tento výpočet výrazne pomalší, predovšetkým preto, lebo sa porovnávajú hodnoty báz sekvencií všetkých atómov. Čas výpočtu teda nezávisí iba od počtu atómov a počtu udalostí, ale aj od dĺžky DNA sekvencie. Aby bolo možné túto metódu použiť v praxi, vôbec nezlepšujeme časovanie udalostí, ale iba vypočítame vierohodnosť s pôvodným načasovaním udalostí, kedy medzi každými dvomi udalosťami uplynie čas 0.01. Vo vyhodnotení porovnávame, či týmto prídeme o presnosť.

3.7 Generovanie susedov

3.7.1 Zvýhodňovanie naposledy použitých udalostí

Prvou metódou, ktorú sme vyskúšali, je pri generovaní rekonštrukcie preferovať udalosti, ktoré sme použili v poslednej akceptovanej rekonštrukcii. Výsledkom by mala byť rekonštrukcia, ktorá sa bude do určitej miery podobať na predošlú, ale nebude vyžadovať, aby boli udalosti použité v rovnakom poradí.

Jediným parametrom tejto metódy je hodnota $prob\_previously\_used\_event$, ktorá určuje minimálnu pravdepodobnosť, že znovu použijeme niektorú z preferovaných udalostí, ak je niektorá z nich k dispozícii.

Implementácia vyžadovala zmeniť spôsob výberu udalostí, ktorý sme opísali v častiach 2.1.1 a 2.1.2. Po navzorkovaní dostaneme pole možných udalostí, pre ktoré vypočítame skóre. Bez zvýhodňovania naposledy použitých udalostí by sme následne náhodne vybrali jednu udalosť, pričom pravdepodobnosť vybratia udalosti je určená pomerom jej skóre k súčtu skóre všetkých navzorkovaných udalostí.

Keďže chceme preferovať udalosti použité v poslednej rekonštrukcii, zvýšime skóre týchto udalostí takým spôsobom, aby sa zachovali pomery skóre medzi nimi, ale záro-

veň bola vyššia pravdepodobnosť, že vyberieme práve jednu z preferovaných udalostí. Označme ako S_{used} súčet skóre preferovaných udalostí a ako S súčet skóre všetkých skóre. Vypočítame pravdepodobnosť, že pri takto ohodnotých skóre vyberieme jednu z preferovaných udalostí:

$$P = \frac{S_{used}}{S}$$

Ak je hodnota P menšia ako $prob\_previously\_used\_event$, vynásobíme skóre každej preferovanej udalosti hodnotou:

$$\frac{prob\_previously\_used\_event}{P}$$

čím zabezpečíme zvýšenie pravdepodobnosti použitia preferovaných udalostí.

Nepopísali sme ešte, ako zistíme, či konkrétna udalosť bola použitá v minulej rekonštrukcii. Udalosti, ktoré získame navzorkovaním, sú uložené v tvare, ktorý zodpovedá ceste po grafe vytvorenom na rýchle vzorkovanie. Namiesto spracovania dát tohto typu ale môžeme znovupoužiť algoritmus na zistenie zmien, ktoré nastali vplyvom udalosti, ktorý popisujeme v časti 4.4. TODO Každú navzorkovanú udalosť aplikujeme, dostaneme v histórii ďalšiu udalosť, zistíme zmeny a

3.7.2 Skopírovanie počiatočných udalostí

Druhým spôsobom generovania podobných rekonštrukcií je vytvoriť novú históriu, ktorá s ňou zdieľa prvých $i - 1$ udalostí, v i-tej udalosti sa líšia a zvyšné udalosti sa samostatne dopočítajú. Na rozdiel od prvej metódy nevyužívame možnosť preusporiadania tých udalostí, pri ktorých je to možné, ale vieme lepšie uvažovať o vlastnostiach výsledných rekonštrukciach.

Dôležitá otázka je, ako voliť pozíciu i? Ak by sme hodnoty i generovali náhodne zo všetkých možných pozícií, robili by sme veľa zbytočnej práce. Čím je totiž i bližšie koncu histórie, o to viac lokálne zmeny nastávajú, lebo väčšia časť histórie je už určená a je možné zmeniť iba posledné udalosti.

Chceme preto začať s hodnotami i, ktoré budú nízke a postupne ich zvyšovať. Najskôr budeme môcť nájsť dobré "základy", na ktorých budeme môcť ďalej stavať a optimalizovať zvyšné udalosti. Proces simulovaného žíhania rozdelíme na tri tretiny - v prvej tretine krokov budeme generovať hodnoty i iba z rozsahu prvej tretiny počtu udalostí aktuálnej rekonštrukcie, v druhej tretine budeme voliť hodnoty z druhej tretiny a v poslednej z tretej tretiny.

Určite by bolo možné nedeliť simulované žíhanie na tri časti, ale napríklad iba na dve alebo naopak viac. Je možné rôzne experimentovať s počtom fáz, na ktoré rozdelíme žíhanie, ale mali sme dojem, že v prípade žíhania trvajúceho 1000 krokov

pripadá na každú fázu 333 krokov, čo pôsobilo dostatočne. V prípade väčšieho počtu fáz a zachovana celkového počtu krokov by sme sa viac sústreďovali na konkrétne rozsahy udalostí v rekonštrukciách, ale zároveň by sme mali menej krokov na ich optimalizáciu. Z toho nám vychádza hodnota 3 ako dobrý kompromis.

Kapitola 4

Porovnanie udalostí a histórií

Aby sme mohli porovnať kvalitu rekonštrukcií rôznych algoritmov, musíme mať spôsob, ako porovnať dve histórie a zistiť, nakoľko sú podobné. V tejto kapitole popíšeme spôsob z minulej práce a nový spôsob, ktorý navrhujeme.

4.1 Pôvodné porovnanie histórií

Pred vysvetlením spôsobu, akým sa v pôvodnej práci porovnávajú histórie, je potrebné vysvetliť, aké údaje sú uložené v dátových štruktúrach reprezentujúcich atómy a udalosti.

Dátová štruktúra atómov neobsahuje iba ich typ, ale aj zoznam identifikátorov atómov, ktoré z nich vznikli duplikáciou. Atómy v súčasnej postupnosti atómov obsahujú v tomto zozname iba svoj identifikátor, ale v prípade duplikácie sa pre atóm, ktorý bol duplikovaný, pridájú aj všetky prvky zo zoznamu atómu, ktorý bol z neho duplikovaný.

V dátovej štruktúre udalosti nie sú uložené menené časti sekvencií, ale musíme ich vypočítať. Pre každú udalosť okrem najstaršej máme k dispozícii postupnosti atómov pred a po jej vykonaní a budeme ich označovať ako $atoms\_A$ (po udalosti) a $atoms\_B$ (pred udalosťou). Ďalej ešte vieme pre každý atóm jeho predka, t.j. atóm z predošlej sekvencie, ktorému zodpovedá. Väčšina atómov je predkom práve jedného atómu, ale duplikované atómy sú predkami dvoch atómov. Túto informáciu máme uloženú v poli, ktoré nazveme *parents*. Má rovnakú veľkosť ako pole atómov $atoms\_A$ a pre i-tý prvok poľa $atoms\_A$ sa na i-tej pozícii v poli *parents* nachádza index predka atómu v poli $atoms\_B$.

V minulej práci sú dve udalosti považované za rovnaké, ak množiny atómov z $atoms\_B$, ktoré majú aspoň dvoch potomkov v $atoms\_A$, sú rovnaké. Aby sa dva atómy v týchto množinách rovnali, musia mať nielen rovnaký typ, ale aj množinu identifikátorov atómov, ktoré z nich neskôr vznikli duplikáciami. V prípade duplikácie "1 2 3" dostaneme množinu 1, 1, 2, 2, 3, 3, v ktorej sa ale jednotlivé atómy s rovnakými

typmi líšia v zozname identifikátorov.

Následne dve histórie sú rovnaké, ak sa v oboch nachádzajú rovnaké udalosti v ľubovoľnom poradí. Nevyžadovanie rovnakého poradia je potrebné, lebo existujú udalosti, ktoré mohli nastať v ľubovoľnom poradí a nevieme rozlíšiť, ktoré z nich je "lepšie" ako iné. Ak by sme napríklad rekonštruovali súčasnú sekvenciu atómov "1 2 3 2 3 4 5 6 5 6 7", duplikácie atómov "2 3" a "5 6" môžu nastať v ľubovoľnom poradí.

Vnímame dva problémy s touto definíciou rovnosti udalostí a histórií. V diplomovej práci je síce definovaná rovnosť v prípade delécií, ale podľa nášho pochopenia existujúceho kódu nie je implementovaná, a nie je zjavné ako ju doimplementovať.

Predovšetkým, nám ale tento prístup umožňuje iba zistiť, či sú histórie ekvivalentné alebo nie, ale v prípade nerovnosti nevieme, nakoľko sa podobajú.

4.2 Nové porovnanie udalostí

Aby sme vyriešili problémy, na ktoré sme poukázali, predstavujeme novú definíciu rovnosti udalostí, z ktorej odvodíme porovnanie histórií.

Duplikáciu alebo duplikáciu s inverziou charakterizuje postupnosť atómov, ktoré pribudli do postupnosti. V prípade duplikácie s deléciami to sú podobne ako v prípade duplikácie postupnosť atómov, ktoré pribudli do postupnosti, a postupnosti atómov, ktoré boli deléciami vymazané. Je dôležité, že v prípade duplikácie s deléciou považujeme tieto dve udalosti za oddelené aj napriek tomu, že sú v pri zobrazení histórie zjednotené v jednej udalosti.

Kvôli zjednodušeniu implementácie nebudeme v prípade atómov porovnávať zoznam identifikátorov atómov, ktoré z nich budú duplikované, ale porovnávame iba typ atómu.

Históriu charakterizuje množina postupností atómov pre všetky udalosti v nej. Ak máme pre dve histórie množiny postupností atómov, ktoré sa počas nich menili, hľadáme metriku, ktorá ich porovná a určí ich podobnosť. Zvolili sme Jaccardov index, ktorý popíšeme v ďalšej časti.

4.3 Jaccardov index

Pre množiny A a B je Jaccardov index (JI) definovaný ako podiel veľkostí prieniku a zjednotenia množín. Hodnotu má v intervale $\langle 0, 1 \rangle$, pričom v prípade prázdnosti oboch množín je dodefinovaný ako 1.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$
$$0 \leq J(A, B) \leq 1$$

Tabuľka 4.1: Používané polia v prípade duplikácie bez delécie

children	[0]	[1,4]	[2,5]	[3]			[6]
atoms_B	1	2	3	4			5
atoms_A	1	2	3	4	2	3	5
parents	0	1	2	3	1	2	4

V prípade, že obe histórie obsahujú totožné udalosti, hodnota JI bude 1. Ak naopak budú histórie veľmi odlišné, potom veľkost prieniku zmenených častí bude malá oproti veľkosti ich zjednotenia a hodnota bude blízko nule.

Jedno z kreatívnych použití Jaccardovho indexu je hodnotenie úspešnosti detekcie objektov v obraze. Ak máme na fotografii nájsť určitý objekt, tak ho môžeme označiť obdĺžnikom, ktorý ho obsahuje. Následne vieme vypočítať pre obdĺžniky správneho a nášho riešenia ich prienik a zjednotenie a vypočítať Jaccardovým indexom mieru ich podobnosti. Úspešné riešenie bude mať zjavne veľký prienik so správnym obdĺžnikom a bude hodnotu JI blízku 1.

4.4 Zistenie zmien

Zistenie zmien sme najskôr realizovali algoritmom na riešenie problému globálneho zarovnania, ale následne sme zistili, že týmto prístupom prichádzame o niektoré užitočné informácie, a našli sme iný spôsob, ktorý dáva lepšie výsledky a navyše pracuje v asymptoticky lepšom čase.

Vytvorme z poľa predkov atómov naopak pole potomkov atómov pre atómy z predošlej sekvencie a označme ho *children*. Je ho možné vidieť spolu so zvyšnými poľami v tabuľke 4.1, ktorá popisuje prípad duplikácie bez delécie. Dôležité je pozorovanie, že duplikovaný úsek ohraničujú atómy, ktoré majú práve dvoch potomkov, a všetky zvyšné atómy s dvomi potomkami sú medzi nimi. Bez ohľadu na typ udalosti nám teda stačí nájsť v poli *children* najľavejší a najpravejší atóm s dvomi potomkami a dostaneme duplikovanú časť. Tieto atómy si nazveme l a r.

Z týchto okrajových prvkov duplikovanej časti takisto vieme zistiť dva rozsahy - odkiaľ bola sekvencia kopírovaná a kam bola vložená. Keďže sú oba rozsahy disjunktné, prvý rozsah zodpovedá menším indexom potomkov l a r a druhý rozsah zodpovedá väčším indexom potomkov. Pre oba rozsahy postupne prechádzame pole *children* medzi atómami l a r a hľadáme atómy, ktoré nemajú potomka v rozsahu. Všetky úseky takýchto atómov zodpovedajú deléciam. Tento prístup umožňuje nájsť aj postupnosti atómov, ktoré boli odstránené z oboch rozsahov. Ukážku použitých polí pre prípad duplikácie a následnej delécie je možné vidieť v tabuľke 4.2.

Túto metódu sme úspešne používali na zistenie zmien v rekonštrukciách, ale keď sme

Tabuľka 4.2: Používané polia v prípade duplikácie s deléciou

children	[0]	[1,5]	[2]	[3,6]	[4]			[7]
atoms_B	1	2	3	4	5			6
atoms_A	1	2	3	4	5	2	4	6
parents	0	1	2	3	4	1	3	5

ju použili na pôvodné generované histórie, nedostávali sme správne výsledky. Problém bol v tom, že generované histórie vytvárali pre delécie samostatné udalosti a neboli teda súčasťou duplikácií. V poli *children* sa to prejavilo postupnosťou atómov, ktoré mali nula potomkov. Upravili sme preto algoritmus, aby zvládal spracovať aj tento druh histórií.

Kapitola 5

Výsledky

V tejto kapitole porovnávame metódy skórovania histórií a generovania susedov a tiež výsledné algoritmy.

5.1 Porovnanie skórovania histórií

V práci používame dve metriky na skórovanie rekonštrukcií, o ktorých predpokladáme, že ich hodnoty súvisia s kvalitou rekonštrukcie. V prípade počtu udalostí očakávame, že rekonštrukcia, ktorá dokáže vysvetliť súčasnú sekvenciu menším počtom udalostí, je lepšia ako rekonštrukcia, ktorá potrebuje viac udalostí. Tiež v prípade merania vierohodnosti histórie očakávame, že história, ktorá má na základe pravdepodobnostného modelu evolúcie väčšiu pravdepodobnosť, že bude vygenerovaná, bude lepšia, ako história s menšou pravdepodobnosťou.

Tieto úvahy sme sa rozhodli overiť. Pre viacero testovacích histórií sme vygenerovali 100 navzájom nezávislých rekonštrukcií a vypočítali pre ne hodnoty oboch metrík a Jaccardovho indexu konkrétnej rekonštrukcie a správnej histórie. Následne sme pre obe metriky vypočítali koeficienty korelácie danej metriky a Jaccardovho indexu a výsledné hodnoty sme uviedli v tabuľke 5.1.

Použili sme Pearsonov korelačný koeficient, ktorý nadobúda hodnoty z rozsahu $\langle -1, 1 \rangle$. 0 znamená neexistenciu lineárnej korelácie, 1 znamená, že zvýšenie jednej premennej spôsobí zvýšenie druhej premennej a -1 znamená, že zvýšenie jednej premennej spôsobí zníženie druhej premennej.

Z výsledkov vidíme, že určitá korelácia medzi metrikami a Jaccardovým indexom existuje, ale nie je veľmi dobrá. Nakoľko ale nemáme lepší spôsob skórovania histórií, musíme sa uspokojiť s týmito. Tiež vidíme, že skórovanie histórií pomocou vierohodnosti nie je lepšie ako výrazne jednoduchšie a rýchlejšie skórovanie pomocou počtu udalostí.

Skúsili sme tiež zlepšiť meranie vierohodnosti tým, že povolíme ešte 10-krát upra-

Tabuľka 5.1: Koeficienty korelácie

História	Počet udalostí	Vierohodnosť
F4100	-0.648812299359216	0.645536048722445
F4104	-0.55880137465962	0.467040509125628
F6100	-0.689568861897211	0.759551291408026
F6101	-0.677955283165815	0.603650307148451

Tabuľka 5.2: Ukážkové dáta skórovaní rekonštrukcií

Počet udalostí	Pôvodná vierohodnosť	Zlepšená vierohodnosť	Jaccardov index
11	-38553.101329	-37778.314696	0.333333
22	-45557.325154	-45539.222203	0.153846
13	-39265.04033	-38913.59407	0.235294
21	-45536.512912	-45484.945861	0.166667
19	-45459.825474	-45377.291759	0.217391
21	-48085.919736	-48028.326658	0.16
9	-36526.941791	-36301.057005	0.5
16	-41368.767623	-40821.19498	0.25
13	-38864.933197	-38845.389593	0.294118
8	-35394.615087	-35194.774907	0.545455

venie časov hrán medzi udalosťami v rekonštrukcií, čím sa máme priblížiť k optimálnej hodnote vierohodnosti. Hodnota 10 je ešte pomerne malý počet zlepšení, lebo pracujeme s rekonštrukciami, ktoré majú aj 30 udalostí, čiže nezlepšíme časy všetkých udalostí a v prípade väčšieho počtu zlepšení by sme dostali lepšiu vierohonosť. Pre históriu *F4100* sme v prípade tejto upravenej metriky vierohodnosti dostali hodnotu koeficientu korelácie 0.648261849340031, čiže sa podarilo zanedbateľne zlepšiť koreláciu, ale čas potrebný na ohodnotenie 100 testovacích rekonštrukcií sa zvýšil zo sekúnd na hodiny, čo robí túto metódu nepoužiteľnú. Domnievame sa ale, že pri úpravách hrán sa často opakovanie počítajú tie isté hodnoty a bolo by možné tento výpočet viac zoptimalizovať.

V tabuľke 5.2 zobrazujeme hodnoty troch metrík a Jaccardovho indexu pre 10 ukážkových rekonštrukcií histórie *F4100*.

5.2 Porovnanie generovania susedov

V práci používame dve metódy na generovanie susedov - buď sme pri výbere udalostí preferovali udalosti použité v poslednej úspešnej rekonštrukcii alebo sme z poslednej udalosti skopírovali určitý počet počiatočných udalostí a zvyšok sme nezávisle dopočí-

Tabuľka 5.3: Hodnoty pomeru znovu použitých udalostí v rekonštrukciách v závislosti od pravdepodobnosti *prob_previously_used_event*

	Pravdepodobnosť			
	0	0.3	0.6	0.9
Minimum	0.25	0.24	0.17	0.16
Maximum	0.72	0.62	0.68	0.65
Priemer	0.40	0.40	0.40	0.36

tali.

Na vyhodnotenie oboch metód sme vždy vygenerovali 100 rekonštrukcií histórie "F6106" a merali, koľko percent udalostí majú totožných s poslednou akceptovanou rekonštrukciou. Nepoužili sme ako metriku Jaccardov index, lebo chceme merať, do akej miery znovu používame minulé udalosti, a nie nakoľko sú rekonštrukcie podobné. Použitím Jaccardovho indexu by bolo problematické hlavne v prípade, keď má sused menej udalostí ako pôvodná rekonštrukcia. Ak by sme napríklad z histórie, ktorá mala 10 udalostí, odvodili suseda, ktorý má 5 udalostí, z toho 3 majú spoločné, tak by pomer znovu použitých udalostí bol $\frac{3}{5} = 0.6$, ale Jaccardov index udalostí pôvodnej histórie a suseda by bol $\frac{2}{13} = 0.15$. Hodnotu teda negatívne ovplyvnili udalosti, ktoré boli v pôvodnej histórií, ale nepoužili sme ich pri generovaní suseda.

Najskôr sa pozrime na výsledky pre prvú metódu v tabuľke 5.3. Očakávali sme, že so zvyšujúcou sa hodnotou *prob_previously_used_event*, ktorá určuje minimálnu pravdepodobnosť, že bude zvolená niektorá z udalostí z poslednej akceptovanej histórie, ak je taká navzorkovaná, sa bude zvyšovať aj miera použitia takýchto udalostí vo vygenerovaných susedoch. To sa ale nestáva, ako vidíme na priemerných hodnotách. Možné vysvetlenie je, že vďaka preferovaniu použitých udalostí sa v niektorom okamihu rekonštrukcie zvolila jedna z použitých udalostí, ktorá ale zmenil sekvenciu atómov takým spôsobom, že zvyšné udalosti z predošlej histórie sa už neskôr nedali použiť. Alebo pri vzorkovaní, čo je náhodný proces, sa nepodarilo nájsť dosť udalostí, z minulej rekonštrukcie.

Výsledky druhej metódy zobrazené v tabuľke 5.4 pôsobia lepšie. Pripomeňme, že táto metóda náhodne zvolí udalosť z pôvodnej histórie, skopíruje udalosť pred ňou do novej histórie, zmení zvolenú udalosť za inú a dopočíta zvyšok histórie. V prvej fáze volí udalosti z prvej tretiny poslednej úspešnej histórie, následne z druhej a na konci z tretej. Aj údaje v tabuľke sú rozdelené podľa fázy. Podľa očakávania s rastúcim číslom fázy sa zvyšuje aj priemerná hodnota pomeru znovu použitia udalostí z minulej udalosti, čo je spôsobené kopírovaním čoraz väčšieho množstva prvotných udalostí. Je zaujímavé, že v prvej fáze kopírujeme síce menej ako tretinu udalostí, ale miera znovu použitia udalostí je v priemere 0.5. To môže byť spôsobené tým, že aj tie udalosti,

Tabuľka 5.4: Hodnoty pomeru znovu použitých udalostí v rekonštrukciách v závislosti od fázy

	Fáza		
	1	2	3
Minimum	0.21	0.59	0.92
Maximum	0.88	1.00	1.00
Priemer	0.50	0.81	0.97

Tabuľka 5.5: Vlastnosti simulovaných histórií s časom generovania 0.04

	avg	**min**	**max**	**F4101**	**F4104**	**F4114**
Počet udalostí	8.3	3	14	3	11	9
Počet atómov	77.3	17	238	17	139	84
Počet typov atómov	24.7	10	41	10	33	25

ktoré neboli skopírované do novej histórie, získali neskôr pri skórovaní dobré skóre a prirodzene vyhrali.

5.3 Vlastnosti simulovaných histórií

Na otestovanie úspešnosti simulovaného žíhania sme vygenerovali 20 simulovaných histórií, ktoré majú čas 0.04, a 20 histórií, ktoré trvajú 0.06. V tabuľkách 5.5 a 5.6 zobrazujeme štatistiky ich vlastností - počet udalostí, počet atómov a počet typov atómov - a tiež tieto údaje pre tri náhodné histórie.

História trvajúce 0.04 sú porovnateľné s najnáročnejšími históriami rekonštruovanými v minulej práci a histórie trvajúce 0.06 sme vygenerovali, aby sme otestovali, ako sa bude simulované žíhanie správať na ešte náročnejších históriach.

5.4 Porovnanie algoritmov

V tejto časti porovnáme algoritmy, ktoré vznikli kombináciou skórovacej funkcie a metódy hľadania susedov, a porovnáme ich výsledky s výsledkami algoritmu z minulej

Tabuľka 5.6: Vlastnosti simulovaných histórií s časom generovania 0.06

	avg	**min**	**max**	**F6105**	**F6106**	**F6119**
Počet udalostí	12.9	7	21	14	11	10
Počet atómov	139.4	46	229	182	112	102
Počet typov atómov	37.8	22	60	43	32	31

práce.

Dostávame nasledujúce algoritmy:

- likelihood-100 - skórovanie podľa vierohodnosti, generovanie susedov pomocou kopírovania počiatočných udalostí, počiatočná teplota 100
- likelihood-prob-0.3 - skórovanie podľa vierohodnosti, generovanie susedov pomocou preferovania udalostí s pravdepodobnosťou 0.3
- likelihood-prob-0.6 - skórovanie podľa vierohodnosti, generovanie susedov pomocou preferovania udalostí s pravdepodobnosťou 0.6
- likelihood-prob-0.9 - skórovanie podľa vierohodnosti, generovanie susedov pomocou preferovania udalostí s pravdepodobnosťou 0.9
- num_events-0.04 - skórovanie podľa počtu udalostí, generovanie susedov pomocou kopírovania počiatočných udalostí, počiatočná teplota 0.04
- num_events-prob-0.3 - skórovanie podľa počtu udalostí, generovanie susedov pomocou preferovania udalostí s pravdepodobnosťou 0.3
- num_events-prob-0.6 - skórovanie podľa počtu udalostí, generovanie susedov pomocou preferovania udalostí s pravdepodobnosťou 0.6
- num_events-prob-0.9 - skórovanie podľa počtu udalostí, generovanie susedov pomocou preferovania udalostí s pravdepodobnosťou 0.9

Následne sme pomocou týchto algoritmov žíhali testovacie histórie 1000 krokov a výsledné rekonštrukcie sme porovnali s rekonštrukciami, ktoré boli výsledkom pôvodného algoritmu. V tabuľkách 5.7, 5.8, 5.9 a 5.10 uvádzame, pre koľko testovacích histórií sme získali lepší, rovnako dobrý alebo horší výsledok.

Z porovnaní nám najlepšie vychádzaja algoritmus *num_events-0.04* a o trochu horšie *num_events-prob-0.9*. Obom sa podarilo vo väčšine prípadov zlepšiť alebo aspoň nezhoršiť výslednú rekonštrukciu.

Tabuľka 5.7: Porovnanie výsledkov algoritmov pre histórie s časom 0.04. Porovnáva sa počet udalostí.

	lepšie	**rovnaké**	**horšie**
likelihood-100	5	3	12
likelihood-prob-0.3	1	13	6
likelihood-prob-0.6	2	8	10
likelihood-prob-0.9	2	4	14
num_events-0.04	9	9	2
num_events-prob-0.3	2	9	9
num_events-prob-0.6	4	6	10
num_events-prob-0.9	4	11	5

Tabuľka 5.8: Porovnanie výsledkov algoritmov pre histórie s časom 0.04. Porovnáva sa Jaccardov index.

	lepšie	**rovnaké**	**horšie**
likelihood-100	8	1	11
likelihood-prob-0.3	8	2	10
likelihood-prob-0.6	9	2	9
likelihood-prob-0.9	7	1	12
num_events-0.04	10	2	8
num_events-prob-0.3	7	1	12
num_events-prob-0.6	6	1	13
num_events-prob-0.9	11	3	6

Tabuľka 5.9: Porovnanie výsledkov algoritmov pre histórie s časom 0.06. Porovnáva sa počet udalostí.

	lepšie	**rovnaké**	**horšie**
likelihood-100	1	3	16
likelihood-prob-0.3	4	1	15
likelihood-prob-0.6	1	4	1
likelihood-prob-0.9	0	0	20
num_events-0.04	11	3	6
num_events-prob-0.3	6	4	10
num_events-prob-0.6	5	6	9
num_events-prob-0.9	4	6	10

Tabuľka 5.10: Porovnanie výsledkov algoritmov pre histórie s časom 0.06. Porovnáva sa Jaccardov index.

	lepšie	**rovnaké**	**horšie**
likelihood-100	3	2	15
likelihood-prob-0.3	8	0	12
likelihood-prob-0.6	7	0	13
likelihood-prob-0.9	4	0	16
num_events-0.04	7	2	11
num_events-prob-0.3	8	3	9
num_events-prob-0.6	6	3	11
num_events-prob-0.9	7	3	10

Kapitola 6

Implementácia a použitie

V tejto kapitole popisujeme implementáciu projektu a zdôrazňujeme viacero problémov a výziev, ktoré sa pri implementácii vyskytli.

6.1 Pôvodný zdrojový kód

Ako základ projektu sme použili zdrojový kód programu HROCH (Heuristic reconstruction of cluster histories), ktorý bol výsledkom práce [4].

V zdrojovom kóde sme opravili viacero chýb, ktoré nás síce značne zdržali, ale aspoň sme pri ich opravovaní boli nútení pochopiť aj kód iných moduloch a lepšie rozumiemeť, ako funguje celý program. Spomenieme aspoň tieto opravené chyby:

- Nebolo možné skompilovať program bez miernych zmien kódu.
- Pri generovaní simulovaných histórií sa neuvolňovala alokovaná pamäť, čo spôsobovalo, že pri generovaní väčšieho množstva histórií sa minula všetka dostupná pamäť.
- Pri invertovaní DNA sekvencie, ktorá obsahovala znak "-" definovaný ako neznáma báza, sa tento znak prepísal na nulový byte, ktorý sa následne nevedel spracovať inými programami.
- V prípade delécií boli zle určované čísla predkov atómov.
- V prípade zjednocovania atómov v stromoch atómov program padal, ak sme sa pokúšali zjednotiť atóm sa sebou samým, pričom táto situácia by nikdy nemala nastať.

Keďže predpokladáme, že tento projekt sa bude ešte dlhodobo vylepšovať a používať, snažili sme sa zlepšiť kvalitu kódu a zjednodušiť ho, aby sa budúci kolegovia mohli sústrediť na implementovanie svojich vylepšení a nie na opravovanie chýb.

- Odstránili sme nepoužívaný kód.
- Na viacerých miestach sme prepísali kód na jednoduchší a ľahšie pochopiteľný.
- Odstránili sme viacero makier. Niektoré ešte zostali, lebo sa používajú z zložitom kóde, ktorému nerozumieme a do ktorého sme radšej nechceli zasahovať.
- Nahradili sme ad hoc parsovanie argumentov programu knižnicou *cxxopts*, ktorá ich umožňuje zadať deklaratívne. Je teda zjednodušené pridávanie nových parametrov, nie je nutné ich zadávať v stanovenom poradí a pri zlom zadaní názvu parametra sa vypíše chyba a program predčasne skončí.
- Pridali sme chýbajúce skripty na vytvorenie stromov atómov a vytvorili nové, napr. na vygenerovanie stromov atómov pre všetky histórie v adresári alebo na spúšťanie veľkého množstva rekonštrukcií s rôznymi parametrami.

6.2 Knižnica na počítanie vierohodnosti rekonštrukcie

Do projektu sme zakomponovali kód, ktorý bol výsledkom práce [1]. Tento program očakáva ako parameter cestu ku konfiguračnému súboru, kde sú určené cesty k súborom s históriou, zoznamom atómov a ich zarovnania. Z programu sme vytvorili statickú knižnicu, ktorej vstupná funkcia dostane ako argumenty tieto parametre. Všetky identifikátory v knižnici sme definovali v samostatnom menovom priestore, aby nedochádzalo ku konfliktom s identifikátormi z nášho projektu.

Aby sme nemuseli každú rekonštrukciu zapisovať na disk a opäť čítať, posielame ju do knižnice v dátovom type "stringstream", čo je dátová štruktúra, do ktorej je možné zapisovať a tiež z nej čítať rovnako ako v prípade súboru na disku, ale dáta si ukladá iba v pamäti. Zoznam atómov a ich zarovnania sa pri každom hodnotení rekonštrukcie čítajú nanovo, ale dá sa očakávať, že tieto dáta budú v pamäti cache a k žiadnemu prístupu na disk nedochádza. Do budúcna by určite bolo užitočné vedieť tieto dáta, ktoré sú statické, načítať iba raz a uložiť ich v dátovej štruktúre.

V programe sme tiež opravili načítavanie zarovnaní atómov, lebo boli ignorované znaky "-" na začiatkoch riadkov.

6.3 Použitie programu

Výstupom práce je program s názvom *hroch.bin*, ktorý zabezpečuje generovanie testovacích simulovaných histórií a rekonštruovanie histórií, a tiež viacero pomocných skriptov.

6.3.1 Pomocné skripty

Perlový skript *sample_trees.pl* slúži na vytvorenie zarovnaní atómov a nájdenie stromu atómov. Používa na to knižnice *shared_dups.pm*, *shared.pm*, ktoré sú súčasťou projektu, tiež musia byť nainštalované knižnica *BioPerl* a programy *muscle* (na zarovnanie) a *MrBayes* (na počítanie stromov). V niektorých prípadoch sa stáva, že tento skript spadne pri volaní programu *MrBayes*, ale nepodarilo sa nám zistiť prečo a jednoducho sme tie histórie ignorovali.

Skript *generate_atom_trees.py* napísaný v jazyku Python 3 voláme bez parametrov z koreňového adresára projektu a pre všetky vygenerované histórie z priečinku "data", ktoré ešte nemajú nájdené stromy atómov, spustí skript *sample_trees.pl* so správnymi parametrami. Tento skript sme napísali, aby sme nemuseli spomenutý skript volať pre každú simulovanú históriu ručne.

6.3.2 Generovanie simulovaných histórií

Parameter "–gen-test" spôsobí vygenerovanie po 200 testovacích histórií s tromi rôznymi hodnotami času simulácie a parameter "–gen-all" vygeneruje okrem týchto histórií ešte 5000 histórií s časom simulácie 0.04 jednotiek. Tieto parametre pochádzajú z pôvodnej práce.

Aby sme nemuseli kvôli generovaniu histórií s inými časmi simulácie meniť zdrojový kód a nanovo kompilovať celý projekt, ale mali flexibilnejší nástroj, pridali sme parameter "–gen", ktorý na základe parametrov "–gen_count", "–gen_prefix" a "–gen_time" vygeneruje daný počet histórií, ktoré sa budú simulovať určený čas a výsledné súbory budú mať zadaný prefix.

Výsledné histórie sa uložia do priečinku "data" a pre históriu s názvom "F1100" sa vytvoria súbory, ktoré spomíname spolu s ich popisom v tabuľke 6.1. Slovo "unicorn", ktoré sa nachádza v názvoch súborov aj vo vnútri súborov, zodpovedá názvu druhu, s ktorým pracujeme, čiže s jednorožcom. V budúcnosti bude možné použiť tento program aj na rekonštrukciu histórií viacerých druhov.

6.3.3 Rekonštrukcia histórií

Na hlavný účel práce, na rekonštruovanie histórií, slúži parameter "–solve". Zvyšné parametre tu nebudeme popisovať, ale je možné ich zobraziť zavolaním programu s parametrom "–help".

Tabuľka 6.1: Štruktúra súborov zodpovedajúcich vygenerovanej histórii s názvom "F1100"

generated-F1100	priečinok obsahujúci súbory ".aln", v ktorých sú zarovnania atómov daných typov atómov
generated-F1100.atoms	zoznam atómov a rozsahov, na ktorých sa nachádzajú
generated-F1100.nhistory	správna duplikačná história
generated-F1100.stats	štatistiky o histórií - počet udalostí, atómov a typov atómov, dĺžka sekvencie a dĺžka zmazaných častí
generated-F1100-unicorn.dna	súčasná sekvencia DNA
dupstemp_generated-F1100-unicorn-dna	priečinok, ktorý vznikne použitím skriptu "sample_trees.pl" a pre každý typ atómu obsahuje v súbore ".aln" zarovanie atómov a v súbore ".trprobs" strom atómov

Záver

V práci sa nám podarilo pomocou použitia simulovaného žíhania podarilo nájsť kratšie alebo rovnako dlhé rekonštrukcie histórií, o ktorých sa domnievame, že sú aj lepšie. Podarilo sa nám to hlavne v prípade histórií trvajúcich čas 0.04, ktoré boli použité aj v minulej práci, ktorá sa venovala tejto téme. V prípade náročnejších histórií trvajúcich čas 0.06 sa nám to tiež podarilo, ale výsledky už boli menej presvedčivé.

Tiež sa nám podarilo popísať a implementovať nový spôsob porovnávania histórií na základe Jaccardovho indexu, ale zistili sme, že korelácia skórovacích funkcií s podobnosťou so správnou rekonštrukciou nie je veľmi veľká.

Za úspech považujeme, že sa nám podarilo prepojiť zdrojové kódy dvoch minulých prác, vďaka čomu môžu budúce vylepšenia počítania vierohodnosti zlepšiť rekonštruovanie histórií.

Do budúcna je možné túto prácu vylepšiť nájdením lepších skórovacích funkcií, ktoré budú lepšie korelovať so Jaccardovým indexom, alebo spôsobu generovania podobných histórií. Takisto nie je implementovaná ešte udalosť speciácia, ktorá spôsobí vznik nového druhu, ktorého DNA sekvencia sa ďalej vyvýja samostatne. Vďaka tejto udalosti by bolo možné rekonštruovať naraz viacero druhov.

Literatúra

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

[2] Broňa Brejová, Martin Kravec, Gad M Landau, and Tomáš Vinař. Fast computation of a string duplication history under no-breakpoint-reuse. *Phil. Trans. R. Soc. A*, 372(2016):20130133, 2014.

[3] Vladimír Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[4] Ján Hozza. Rekonštrukcia histórií génových zhlukov. Master's thesis, Comenius University in Bratislava, 2016. Supervised by Tomáš Vinař.

[5] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

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

[7] Tomáš Vinař, Broňa Brejová, Giltae Song, and Adam Siepel. Reconstructing histories of complex gene clusters on a phylogeny. *Journal of Computational Biology*, 17(9):1267–1279, 2010.