Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vizualizácia evolučných histórií

Bakalárska práca

2016
Dávid Simeunovič

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vizualizácia evolučných histórií

Bakalárska práca

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

Bratislava, 2016
Dávid Simeunovič

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

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

Názov:

Cieľ:

Literatúra:

Kľúčové slová:

Vedúci:
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania:

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

.....
študent

.....
vedúci práce

Poďakovanie:

Abstrakt

Počas evolúcie dochádza v DNA k lokálnym mutáciám, ktoré menia jeden alebo niekoľko susedných nukleotidov, ale aj k väčším zmenám, ktoré menia poradie alebo počet výskytov dlhších oblastí. Cieľom práce je implementovať systém na vizualizáciu evolučnej histórie jednej alebo viacerých DNA sekvencií s dôrazom na tieto väčšie zmeny. Samotná história je daná na vstupe a cieľom je zobraziť ju tak, aby sa dali prehľadne sledovať jednotlivé mutácie a tiež vzťahy rôznych častí sekvencie.

Kľúčové slová: vizualizácia,evolučná história,poradie génov

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Obsah

Zoznam obrázkov

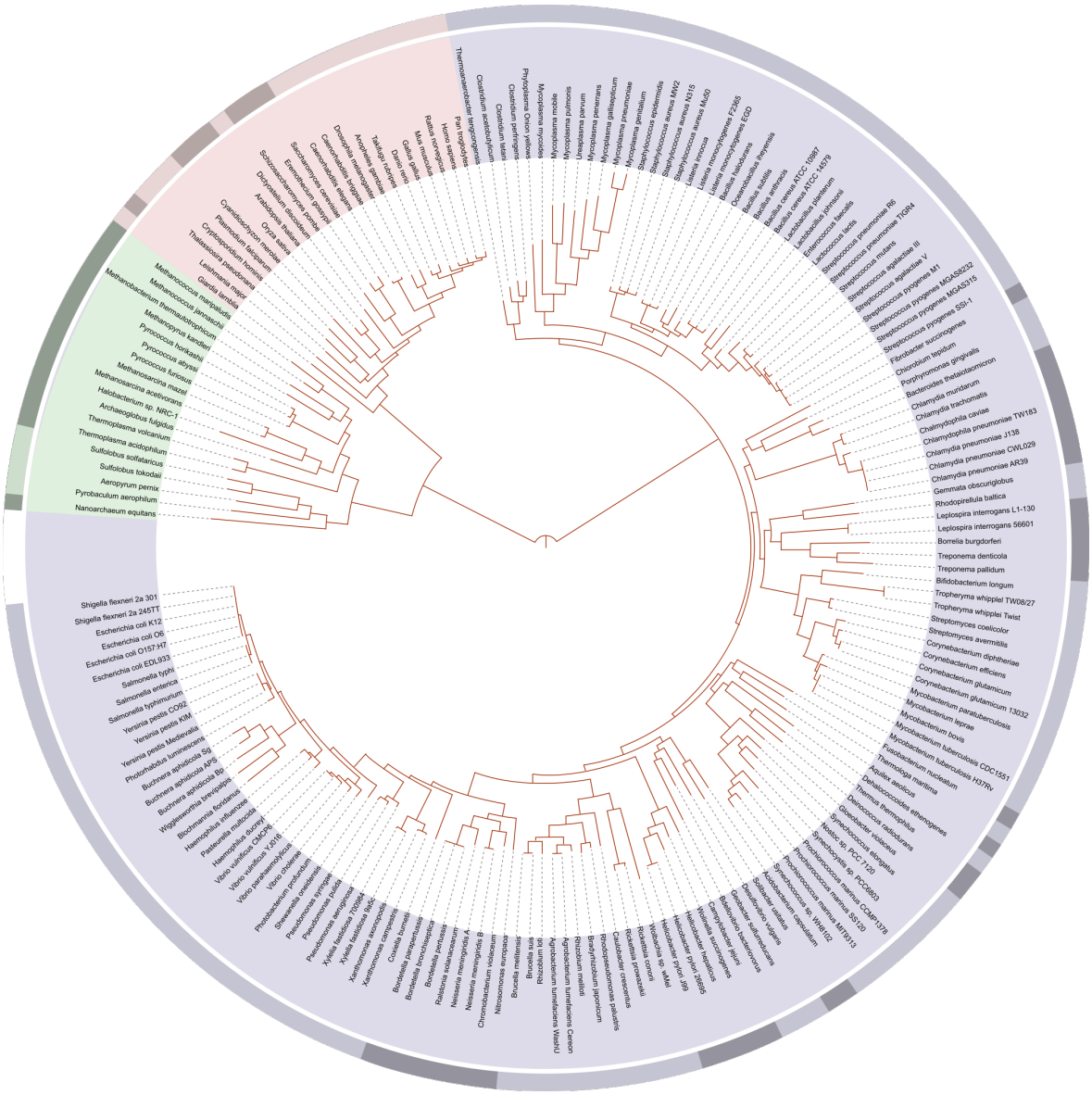
Úvod

Už v antike sa niektorý antický folozofi zaoberali myšlienkou, že základ jendotlivých druhov sa časom mení. Neskôr v tomto vednom poli urobil významný pokrok Charles Darwin keď v roku 1859 publikoval svoju knihu *Pôvod druhov*, okrem toho, že z evolúcie vytvoril široko uznávanú teóriu, predstavil aj myšlienku spoločného predka, kedy akékoľvek dva veľmi rozdielne druhy zdieľajú spoločného prapredka a vizuálne ju znázornil vo forme stromového grafu, takzvaného *stromu života*. Tak položil základy Evolučnej histórie, ktorá skúma evolučné procesy, ktoré na zemi vytvorili rôznorodosť života z počiatočnej živej formy. Ďalšie poznatky v oblasti genetiky, súvisejúce s DNA a RNA viedli k tomu, že na evolúciu sa dodnes pozeráme hlavne prostredníctvom génov. Sekvenovanie DNA umožinilo vzťahy medzi jednotlivými organizmami odsledovať na základe rozdielnosti ich DNA sekvencie. Aj na našej fakulte vzniklo niekoľko prác, ktoré sa venujú rekonštrukcii DNA sekvencie pokiaľ poznáme jej súčasný vzhľad, prípadne sa snažia zrekonštruovať fylogenetický strom pokiaľ poznáme DNA sekvencie súčasných druhov [4, 2, 1, 7].

Strom stále patrí medzi najpopulárnejšie spôsoby ako zobraziť evolučné vzťahy medzi druhmi alebo inými objektami. Najčastejšie sa stretávame so stromom života, kedy je snaha zobraziť vývoj druhov z posledného spoločného prapredka, ako napríklad vidíme na obrázku 1. V týchto prípadoch sa samotné zmeny DNA do zobrazenia nezvyknú dostať, informácie ktoré nám poskytnú, ako napríklad vzdialenosť dvoch objektov na základe rozdielnosti ich DNA sekvencie, však bývajú použité na zostavenie takéhoto zobrazenia. Cieľom tejto práce je zostrojiť program ktorý nám zobrazí jednoduchú postupnosť sekvencí DNA s dôrazom na zmeny ktoré sa udiali s génmy v týchto sekvenciách. Výsledok by mal predstavovať malú vetvu fylogenetického stromu v ktorom prepojenie objektov zobrazí reálne zmeny ktoré sa odohrali na ich DNA sekvencii. Inšpiráciou pre túto prácu sú vyššie spomenuté práce pochádzajúce z našej fakulty, náš program má byť schopný vizualizovať výsledky ktoré produkujú a poslúžiť okrem iného ako rýchla optická kontrola správnosti.

Prvá kapitola nám poskytne úvod do problemtaiky, predstavíme si základné pojmy potrebné pre našu prácu.

Druhá kapitola sa bude venovať implementácii nášeho programu, pozrieme sa na to ako vyzerá jeho vstup a výstup, aké možnosti interakcie poskytuje užívateľovi a ktoré



Obr. 1: Strom života, Zdroj: wikipedia.org

nastavenia v ňom vieme meniť.

Tretia kapitola nám povie prečo chceme zobraziť iba niektoré gény a akým spôsobom ich budeme vyberať.

Štvrtá kapitola nadväzuje na tretiu, porovnávame v nej aké výsledky dostaneme v závislosti od toho aký spôsob výberu génov zvolíme.

Kapitola 1

Úvod do problematiky

V tejto kapitole si vysvetlíme základne pojmy potrebné pre túto bakalársku prácu.

1.1 Biologické pozadie

1.1.1 DNA,Gén,Genóm

DNA Deoxyribonukleová kyselina je nositeľom genetickej informácie bunky.
Má štruktúru dvojzávitnice, skladajúcej sa z dvoch komplementárnych vlákien. Vlákno je tvorené nukleotidmy, ktoré obsahujú jednu zo štyroch báz Adenín,Guanín,Tymín a Cytozín. DNA zvykneme zapisovať ako postupnosť týchto báz, kde každú bázu kódujeme jej počiatočným písmenom - A,G,T,C.

Gén Gén je súvislý úsek DNA ktorý kóduje tvorbu proteínu. Gén je základnou jednotkou dedičnosti.

Genóm Genóm je súbor DNA molekúl organizmu, ktoré sa väčšinou nachádzajú v chromozómoch. Napríklad v ľudskom tele sa jedná o 46 molekúl DNA, jedna v každom chromozóme[6].

1.1.2 Evolučná história

Evolučná história je postupnosť udalostí, ktoré sa odohrali na nejakej DNA sekvencii. Pre potreby tejto práce sú podstatné udalosti odohrávajúce sa na dlhých úsekoch DNA - génoch.
Možné udalosti sú:

- *Duplikácia* - skopírovanie génu na iné miesto v DNA.

- *Inzercia* - vloženie nového génu.
- *Delécia* - odstránenie génu.
- *Inverzia* - zmena poradia a orientácie génu alebo génov.
- *Translokácia* - zmena poradia génu alebo génov
- *Speciácia* - špeciálna udalosť, ktorá označuje vznik nového druhu. Vzniká nová vetva v evolučnej histórii.

Krok evolučnej histórie

Krok evolučnej histórie, ďalej len *krok e.h* je pre nás známa sekvencia génov. Medzi jednotlivími krokmi došlo k jednej alebo viacerím udalostiam.

1.1.3 Fylogenetický strom

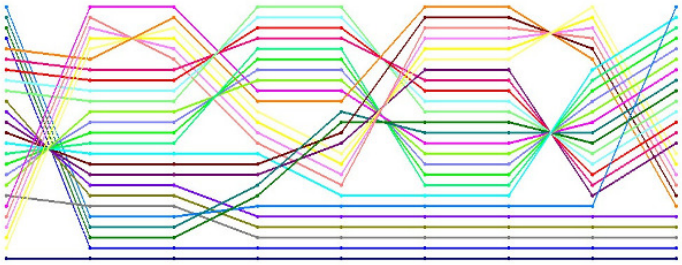
Fylogenetický strom je grafické znázornenie, ktoré zobrazuje evolučné vzťahy medzi sadou objektov. Pokiaľ si za objekty zvolíme druhy, jedná sa o takzvaný *Druhový strom*. Jednotlivé druhy sú pospájané hranami, ktoré reprezentujú evolučný vzťah.
Druhy, ktoré sa nachádzajú na *listoch* stromu sú buď existujúce druhy, z ktorých sa nevyvinuli nové druhy, alebo vyhynuté druhy bez potomkov.
Vnútorné vrcholy predstavujú predchodcov, o ktorých sa predpokladá že sa vyskytli počas evolúcie.
Pokiaľ je v strome známy posledný spoločný predok, nazveme ho *koreň*, a takýto strom označíme ako *zakorenený*.
V *zakorenenom* strome je zrejmá orientácia vnútorných hrán, ktorá určuje ktorý druh sa vyvinul z ktorého.

1.2 Vizualizácia

Vizualizácia je spôsob prevodu dát do grafickej formy, ktorú vieme spracovať naším zrakom, najdominantnejším zmyslom aký máme. To nám umožnuje okrem lepšieho pochopenia problému aj rýchlu analýzu a odhalenie existujúcich súvislostí a vzorov ktoré sa nachádzajú vo výsledku.

1.2.1 Iné programy

Iné programy použitelné pre vizualizáciu fylogenetických stromov, ako napríklad *phylo.io*,*ETE toolkit* alebo *Archaeopteryx* neponúkajú funkcionalitu ktorú potrebujeme. Poskytujú vizualizáciu fylogenetických stromov v ktorých gény buď vôbec nevystupujú alebo sa



Obr. 1.1: Výstup z programu Michaely Sandalovej, Zdroj: [6]

nachádzajú iba pri listoch stromu, nebýva na nich však zobrazený vzťah medzi jednotlivými vrcholmi stromu. Rozdiely zviknú byť vyjadrené číselne ako vzdialenosť genómov. Najväčší dôraz sa kladie na vetvenie a listy stromu, vrcholy nachádzajúce sa vo vnútri stromu slúžia len ako miesta pre rozvetvenie. V našej práci chceme zmeny medzi vrcholmy stromu zobraziť práve pomocou udalostí ktoré sa odohrali. Vetvenie pre nás nieje kľúčové Naša práca je nasledovníkom programu Michaely Sandalovej, ktorý zobrazoval preusporiadania postupnosti génov - obrázok 1.1.

Kapitola 2

Implementácia programu

V tejto kapitole sa budeme venovať niektorým významnejším črtám implementácie nášho programu, ktorý na vstupe dostane súbor popisujúci evolučnú históriu, umožní uživateľovi zmeniť niektoré nastavenia a prípadne spustiť optimalizáciu a nakoniec zobrazí grafickú reprezentáciu vstupnej evolučnej histórie podľa podľa toho aké zmeny vykonal uživateľ. Bližšie sa pozrieme na to v akom formáte má byť zapísaný vstup, ako bude vyzerať výstup, aké kroky vykoná náš program, akým spôsobom dokáže uživateľ interagovať s programom, ako aj to ktoré nastavenia vieme meniť a čo reprezentujú.

2.1 Vstup

Vstup musí obsahovať dáta, ktoré nám umožnia vykresliť fylogenetický strom, a zobraziť v ňom k akým zmenám v genóme došlo, a ktoré udalosti sú za to zodpovedné. To znamená že v infromáciách o jednotlivých vrcholov sa budú nachádzať aj poznatky o tom, ako vyzerá genóm daného vrcholu, a aké sú vzťahy medzi týmto a genómom jeho predchodcu.

2.1.1 Formát

Vstupný súbor nášho programu bude tvoriť postupnosťou riadkov, podobná tej akú vidíme v tabulke 2.1.1. Každý riadok predstavuje jeden *krok e.h.*. Prvý riadok je koreňom danej histórie, opisuje prvotný stav sekvencie a má priradenú špeciálnu udalosť "root". Každý ďalší riadok, opisuje niektorý z nasledujúcich krokov e.h. obsahuje zoznam génov nachádzajúcich sa v tomto kroku a spolu so svojím predchodcom nám umožnuje určiť aké udalosti z 1.1.2 viedli k súčasnému stavu. Predchodca sa v súbore musí nachádzať vždy skôr než nasledovník, aj preto je prvým riadkom koreň.
Riadok obsahuje niekoľko reťazcov a čísel, oddelených medzerou alebo viacerými medzerami, ak je to potrebné pre lepšiu prehľadnosť.

predok	e1	root	0	root	1 2 1 5 4 3 2	#	-1 -1 -1 -1 -1 -1 -1
predok	e2	e1	0.05	dup	1 2 1 2 5 4 3 2	#	0 1 2 1 3 4 5 6
clovek	e3	e2	0.12	sp	1 2 1 2 5 4 3 2	#	0 1 2 3 4 5 6 7
clovek	e4	e3	0.13	del	1 2 1 2 4 3 2	#	0 1 2 3 5 6 7
clovek	e5	e4	0.14	ins	1 2 1 6 7 2 4 3 2	#	0 1 2 -1 -1 3 4 5 6
clovek	e6	e5	0.2	inv	1 -1 -2 6 7 2 4 3 2	#	0 2 1 3 4 5 6 7 8
clovek	e7	e6	0.25	leaf	1 -1 -2 6 7 2 4 3 2	#	0 1 2 3 4 5 6 7 8
simpanz	e8	e2	0.12	sp	1 2 1 2 5 4 3 2	#	0 1 2 3 4 5 6 7
simpanz	e9	e8	0.2	leaf	1 2 1 2 5 4 3 2	#	0 1 2 3 4 5 6 7

Tabuľka 2.1: Ukážka vymysleného vstupu v súčasnom formáte [6]

Význam stĺpcov:

Prvý stĺpec je názov objektu, ktorého sa týka daný riadok.

Druhý stĺpec je id riadku.

Tretí stĺpec je id predchodcu, prvý riadok má špeciálne id predchodcu s hodnotou root.

Štvrtý stĺpec je čas, v ktorom sa daná udalosť odohrala. Koreň sa nachádza v čase 0, a čas je rastúci.

Piaty stĺpec je skratka niektorej z udalostí, popísaných v sekcii 1.1.2 pokiaľ sa v danom *kroku e.h.* odohrala iba jedna z týchto udalostí, alebo jedna zo trojice udalostí root/leaf/other. Root je udalosť slúžiaca na identifikáciu koreňa. Leaf slúži na určenie času v ktorom sa daná vetva končí, medzi udalosťou označenou ako leaf a jej predchodcom nemuselo prísť k žiadnym zmenám. Other použijeme ak tento rozdiely medzi týmto a predchádzajúcim krokom niesme schopný popísať pomocou jednej udalosti, znamená to že takýto krok vznikol kombináciou viacerých udalostí ako napríklad dve duplikácie nasledujúce po sebe alebo translokácia s následnou deléciou.

Nasledujúce stĺpce obsahujú postupnosť génov. Každý gén je celé číslo, pričom znamienko určuje jeho orientáciu. To znamená že gén 2 je rovnaký ako gén -2, iba opačne orientovaný v rámci dna.

Znak # slúži ako ukončenie zoznamu génov.

Zvyšné stĺpce pre každý gén určujú poradie predka génu v predchodcovi jeho riadku. Ak tento gén nemá predchodcu, obsahuje riadok hodnotu -1. Napríklad pre gén 2

z riadku e4 vieme, že poradie jeho predchodcu má index 3. Keďže predchodcom e4 je e3 vieme spojiť štvrtý (indexujeme od nuly) gén z riadku e3 s štvrtým génom (gén 2) riadku e4.

predok	e1	root	0	root	1 2 1 5 4 3 2	#	-1 -1 -1 -1 -1 -1 -1
predok	e2	e1	0.05	dup	1 2 1 2 5 4 3 2	#	0 1 2 1 3 4 5 6
clovek	e3	e2	0.12	sp	1 2 1 2 5 4 3 2	#	0 1 2 3 4 5 6 7
clovek	e4	e3	0.13	del	1 2 1 2 4 3 2	#	0 1 2 3 5 6 7
clovek	e5	e4	0.14	ins	1 2 1 6 7 2 4 3 2	#	0 1 2 -1 -1 3 4 5 6
clovek	e6	e5	0.2	inv	1 -1 -2 6 7 2 4 3 2	#	0 2 1 3 4 5 6 7 8
clovek	e7	e6	0.25	leaf	1 -1 -2 6 7 2 4 3 2	#	0 1 2 3 4 5 6 7 8
simpanz	e8	e2	0.12	sp	1 2 1 2 5 4 3 2	#	0 1 2 3 4 5 6 7
simpanz	e9	e8	0.2	leaf	1 2 1 2 5 4 3 2	#	0 1 2 3 4 5 6 7

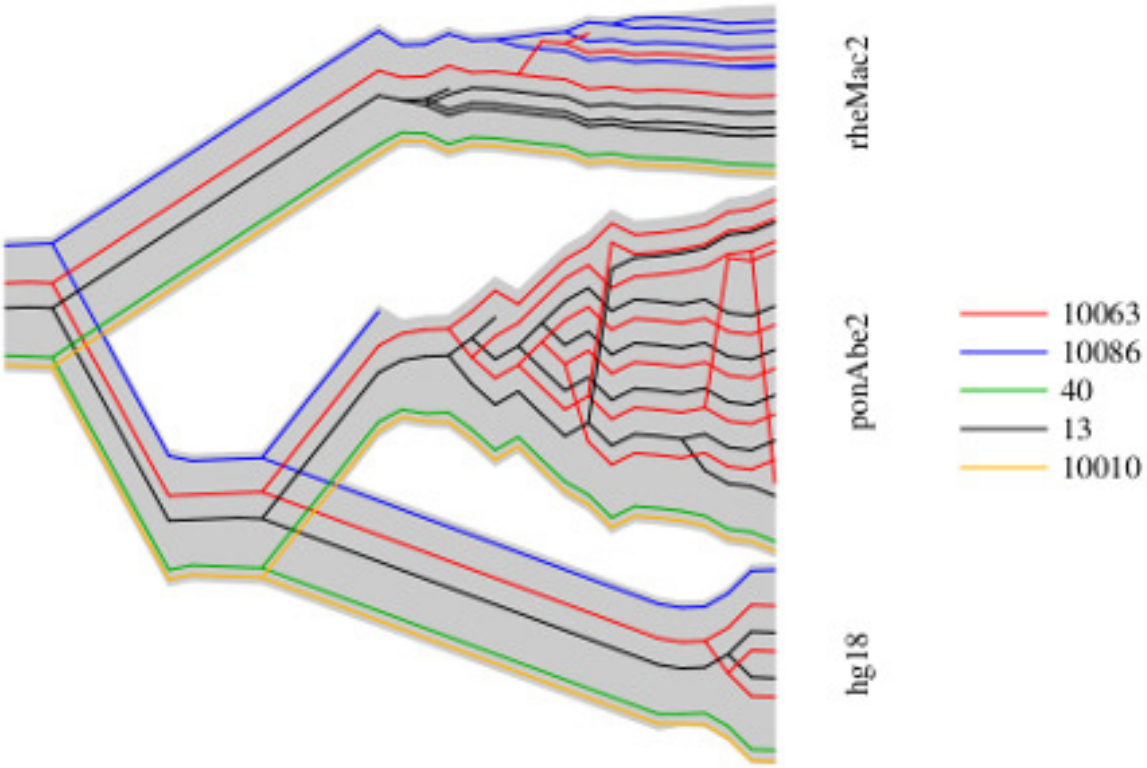
Tabuľka 2.2: Ukážka vymysleného vstupu v súčasnom formáte [6]

2.2 Návrh výstupu

Výstupom programu je obrázok 2.1 zakoreneného fylogenetický stromu , ktorý zobrazuje evolučné vzťahy medzi rôznymi druhmi na základe vzťahov medzi ich génmi. X-ová os reprezentuje čas, v ktorom sa jednotlivé udalosti odohrali.
Strom druhov slúži ako pozadie pre gény.
Gény sú znázornené farebnými čiarami, ktoré idú vodorovne až kým nenastane nejaká udalosť.
Duplikácia je znázornená rozvetvením génu.
Speciácia rozvetvením všetkých génov, a na rozdiel od duplikácie sa vetví aj strom druhov.
Inzercia génu je znázornená ako pridanie novej čiary, na prislúchajúce miesto do stromu druhov.
Delécia je ukončenie čiary, ktorá znázorňuje gén.
Inverzia je znázornená ako kríženie čiar preusporiadaných génov.
Leaf je znázornení ako ukončenie stromu druhov a všetkých génov v tejto vetve.
Root je začiatok stromu druhov a aj všetkých génov, ktoré sa nachádzajú v počiatočnom predkovi.

2.2.1 Možné zmeny

Súčasný návrh vizualizácie nezobrazuje všetky informácie zo vstupu. Jedným z údajov, ktorý sa stráca je orientácia génu.



Obr. 2.1: Možný vzhľad fylogenetického stromu [7]

Kapitola 3

Problém množinového pokrytia a výber génov

V predchádzajúcej kapitole sme si predstavili základné prvky nášho programu ktorý dostane na vstupe súbor, popisujúci evolučnú históriu a na výstupe nám vykreslí fylogenetický strom reprezentujúci danú históriu. Problém nastane, pokiaľ v histórii nachádza príliš veľa génov. Výsledný vygenerovaný obrázok sa stáva neprehľadným, a získanie informácie z neho obtiažne. Potrebujeme teda vybrať iba niektoré gény na zobrazenie tak, aby na obrázku zostali zachované podstatné informácie. V tejto kapitole si predstavíme spôsob, akým budeme vyberať ktoré gény zobrazíme, využitie *Problému množinového pokrytia* pri hľadaní daných génov a dva algoritmy ktoré riešia daný problém.

3.1 Výber génov

Najpodstatnejšou informáciou pri analýze fylogenetického stromu je pre nás to, aké udalosti sa v ňom odohrali. Budeme sa teda snažiť nájsť podmnožinu všetkých génov tak, aby všetky udalosti ostali na obrázku zachované. Zvyšné gény následne z obrázku odstránime, čo môže viesť k strate informácií ktoré považujeme za menej podstatné, ako napríklad to, koľko a ktoré geny sa nachádzajú v danej histórii, ako aj koľko a ktoré gény sú ovplyvné danou udalosťou.

3.1.1 Blok

Blok predstavuje postupnosť génov ktoré sa pred aj po kroku e.h. nachádzali vedľa seba v rovnakom poradí a jednotlivé gény nemenili svoju orientáciu. Jedná sa teda o súvislý úsek DNA ktorý počas kroku e.h. nebol prerušený. Ak sa pri delécii alebo inzercii odobralo alebo pridalo viacero génov, a nenachádza sa medzi nimi žiaden iný gén, tvoria jeden blok. Pri duplikácii gény tvoria blok ak sa nachádzali pri sebe pred

duplikáciou a rovnako aj po nej vo všetkých zduplikovaných inštanciách. Pri Inverzii sa gény nachádzajú v bloku pokiaľ sa všetkým zmení orientácia, t.j. zrotuje celý blok. Napr blok génov (4,5,-6) bude po inverzii vyzerať ako (6,-5,-4).

Pokrytie blokov

Blok považujeme za pokrytý pokiaľ sa na obrázku vyskytuje aspoň jeden gén patriaci do daného bloku. Pokrytie všetkých blokov jedného kroku e.h nám zaručí zobrazenie všetkých udalostí, aj keď nie v úplnom rozsahu, ktoré sa v danom kroku vyskytli. Musíme preto nájsť také gény, ktoré pokryjú všetky bloky v kompletnej evolučnej histórii, a tým si zaistiť zobrazenie všetkých udalostí vo výslednom fylogenetickom strome. Ako cieľ si zvolíme aby bola daná množina génov čo najmenšia.

3.2 Problém Množinového Pokrytia

Definícia Máme dané univerzum$U$, ktoré obsahuje n prvkov, a systém jeho podmnožín $S = \{P_i : P_i \subseteq U\}$, ktorý pokrýva celé univerzum $\cup_{Pi \epsilon S} Pi = U$, vybrať čo najmenšiu množinu podmnožín $C \subseteq S$ takú ktorá tiež pokryje celé Univerzum $\cup_{Pi \epsilon C} Pi = U$. Problém Množinového Pokrytia (anglicky Set Cover Problem), ďalej len *PMP*. patrí medzi NP-úplne problémy.[3]

Príklad Pre Univerzum$U = \{1, 2, 3, 4, 5, 6\}$
a systém jeho podmnožín $S = \{\{1, 2, 3\}, \{2, 3\}, \{3, 4\}, \{3, 4, 6\}, \{5\}\}$
je riešením množina podmnožín $C = \{\{1, 2, 3\}, \{3, 4, 6\}, \{5\}\}$

3.2.1 Výber génov pomocou Problému Množinového Pokrytia

Výber takých génov ktoré pokryjú všetky bloky v celej evolučnej histórii vieme formulovať ako Problém Množinového Pokrytia Univerzum predstavuje všetky bloky ktoré sa nachádzajú v našom fylogenetickom strome. Každý gén predstavuje jednu podmnožinu, v ktorej sa nachádzajú tie bloky, cez ktoré gén prechádza. Riešením je taká množina génov, ktorých zjednotenie pokrýva všetky prvky Univerza, v našom prípade všetky bloky nachádzajúce sa v evolučnej histórii.

3.3 Riešenie Problému Množinového Pokrytia

Keďže *PMP* patrí medzi NP-ťažké problémy, znamená to že zatiaľ neexistuje, a možno nikdy ani nebude existovať algoritmus ktorý by dokázal nájsť riešenie v polynomiálnom čase. Potrebujeme sa teda rozhodnúť, či je pre nás výhodnejšie hľadať najlepšie riešenie *PMP* čo môže byť časovo náročné, alebo sa uspokojíme s približním riešením, ktoré

sme schopný nájsť aproximačným algorimtom v polynomiálnom čase, a ktoré môže taktiež predstavovať dostatočné odstránenie prebytočných génov z obrázku. Predstavíme si jeden spôsob ktorým budeme hladať úplné riešenie, jeden spôsob na nájdenie približného riešenia a v nasledujúcej kapitole porovnáme výsledky ktoré produkujú.

3.3.1 Greedy algoritmus

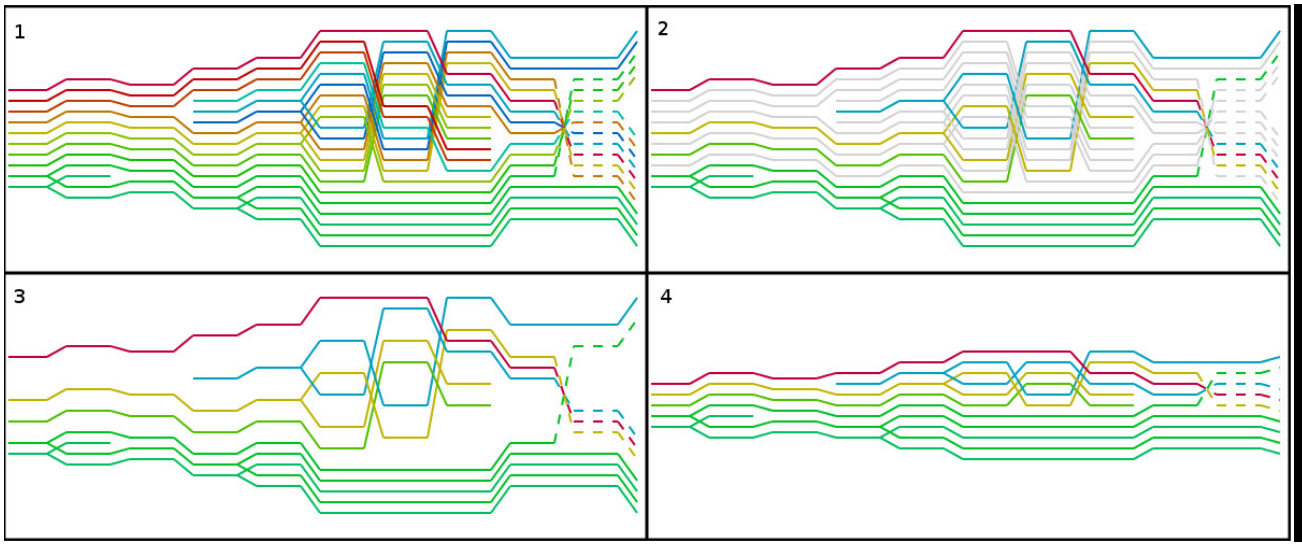
Greedy algoritmus patrí medzi najlepšie polynomiálne aproximačné algoritmy pre riešenie *PMP*.[5] Greedy algoritmus v každom kroku pridá do riešenia takú podmnožinu, ktorá obsahuje najviac zatiaľ nepokrytých prvkov univerza. Riešenie teda hľadáme nasledovným spôsobom:
Všetky podmnožiny zoradíme na základe toho, koľko prvkov obsahujú. Do riešenia vyberieme najvačšiu podmnožinu a prvky, ktoré sa v nej nachádzajú odstránime z univerza aj zo zvyšných podmnožín. Zvyšné podmnožiny opäť zoradíme podľa veľkosti, a postup opakujeme až pokým nie je Univerzum prázdne.
Tesná analýza podľa Slavíka ukazuje, že aproximačný koeficient takéhoto riešenia je $\ln m - \ln \ln m \ + \Theta(1)$ [5] kde $m = |U|$.

3.3.2 Binárne-Celočíselné Lineárne Programovanie

Lineárne programovanie je optimalizačná úloha, pri ktorej je cieľom nájsť minimum alebo maximum lineárnej funkcie f s n-premennými, zatiaľ čo máme dané lineárne obmedzenia vo forme rovníc a nerovníc. V prípade binárneho-celočíselného lineárneho programovania nadobúdajú premenné hodnotu 0 alebo 1, a všetky atribúty obmedzujúcich rovníc a nerovníc sú celočíselné. Binárne-Celočíselné lineárne programovanie (angl: 0-1/binary integer linear programming),ďalej BILP patrí medzi NP-úplné problémy.[3] Množstvo iných problémov, ako napríklad Problém obchodného cestujúceho, Problém Vrcholového pokrytia a *PMP* môžu byť formulované ako Celočíselné linárne programovanie. Navyše pre Celočíselné lineárne programovanie existuje množstvo

Prevedenie výberu génov na BCLP

Všetky gény nachádzajúce sa v našej evolučnej histórii očíslujeme číslom $1 - n$, premenná x_i bude nadobúdať hodnotu 0 alebo 1 v závislosti od toho či sa gén pod číslom i nachádza v riešení. Lineárna fukncia ktorú cheme minimalizovať bude v tvare $min\ x_1 + x_2 + x_3 + .. + x_n$. Lineárne obmedzenia vytvoríme tak, že pre každý blok B_a sa pozrieme na všetky gény ktoré daný blok pokrývajú x_{ai} : $x_{ai} \epsilon B_a$, a pridáme podmienku že súčet premmených x_{ai} reprezentujúcich takéto gény musí byť väčší ako 1, $\{x_{a1} + x_{a2} + .. + x_{aj} \geq 1 | \forall i_{i \epsilon \{1..j\}} : x_{ai} \epsilon B_a\}$, to znamená že v riešení sa musí nachádzať aspoň jeden gén pokrývajúci daný blok. Vo výslednom *CLP* sa bude nachádzať



Obr. 3.1: Kroky optimalizácie

jedna lineárna funkcia ktorú chceme minimalizovať, tá obsahuje n premenných kde $n =$ počet génov $= |S|$. Plus k lineárnych obmedzení, kde $k =$ počet blokov $= |U|$.

Riešenie BCLP Cieľom tejto práce nie je nájsť najlepší spôsob, alebo zostrojiť najlepší program, pre riešenie *BCLP* či *PMP*. Oba problémy sú len prostriedkom ako dosiahnuť optimalizáciu množstva zobrazených génov a tým zvýši prehladnosť. V prípade greedy algoritmu sa implementácia nachádza priamo v našom programe, čo nám umožnuje v krátkom čas dospieť aspoň k čiastkovej optimalizácii. V prípade BCLP náš program nevie nájsť riešenie, ponúka však možnosť vyexportovať sformulovaný *BCLP*. Pre samotné riešenie *BCLP* je vhodné použit niektorý z existujúcich nástrojov[8], a riešenie nahrať do nášho programu. pre účely tejto práce bol využívaný IBM ILOG CPLEX Optimization Studio - *CPLEX*

3.3.3 Výsledok optimalizácie

Výsledkom optimalizácie génov si môžme ilustrovať na obrázku 3.1. Na začiatku dostane náš program evolučnú históriu s kompletnou informáciou, ako vidíme v časti 1). Následne zostrojíme všetky bloky a nájdeme riešenie pre daný *PMP*. V časti 2) sú gény, ktoré sa nachádzajú v riešení vyznačené farebne, zvyšné gény sú šedé. Prebytočné gény z obrázku odstránime. V časti 3) môžme pozorovať, ako sa ich odstránením obrázok preriedi, napriek tomu ostávajú všetky udalosti zachované. Časť 4) je finálnym krokom optimalizácie, prebytočné gény už viac nezaberajú žiadne miesto, všetky udalosti zostali zachované, nie sme však už schopný určiť ich pôvodný rozsah, ani množstvo génov ktoré sa kedysi nachádzali na obrázku.

Záver

V závere je potrebné v stručnosti zhrnúť dosiahnuté výsledky vo vzťahu k stanoveným cieľom. Rozsah záveru je minimálne dve strany. Záver ako kapitola sa nečísluje.

Literatúra

[1] Albert Herencsár. An improved algorithm for ancestral gene order reconstruction. Master's thesis, Comenius University in Bratislava, 2014. Supervised by Broňa Brejová.

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

[3] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[4] Jakub Kovac, Brona Brejova, and Tomas Vinar. A Practical Algorithm for Ancestral Rearrangement Reconstruction. 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 163–174, Saarbrücken, Germany, September 2011. Springer.

[5] Petr Slavík. A tight analysis of the greedy algorithm for set cover. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 435–441, New York, NY, USA, 1996. ACM.

[6] Tomas Vinar and Brona Brejova. Biowiki, 2014.

[7] 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.

[8] Wikipedia. Linear programming — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Linear%20programming&oldid=713865251`, 2016. [Online; accessed 10-May-2016].

[9] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2011.

[10] M.J. Zvelebil and J.O. Baum. *Understanding Bioinformatics*. Garland Science, 2008.