UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

IDENTIFIKÁCIA VARIANTOV GÉNOV Z DÁT SEKVENOVANIA NOVEJ GENERÁCIE

2013 **Martin BOBÁK**

Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Identifikácia variantov génov z dát sekvenovania novej generácie

(Diplomová práca)

Martin Bobák

Študijný program: Informatika
Študijný odbor: 9.2.1. Informatika (2508)
Konzultant: Mgr. Martin Kravec
Školiteľ: Mgr. Tomáš Vinař, PhD.

Bratislava, 2013

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Martin Bobák
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Identifikácia variantov génov z dát sekvenovania novej generácie

Cieľ: Najnovšie technológie na sekvenovanie DNA produkujú veľké množstvá dát často útržkovitej povahy. Cieľom je navrhnúť metódu, ktorou možno z takýchto dát zostaviť varianty génov, ktoré sa v genóme nachádzajú vo viacerých kópiách, pričom tieto kópie nemožno od seba spoľahlivo odlíšiť pomocou štandardných metód založených na deBruijnových grafoch.

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

Dátum zadania: 14.10.2011

Dátum schválenia: 24.10.2011 prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce,
konzultant

Poďakovanie

Týmto by som sa chcel poďakovať Tomášovi Vinařovi, Martinovi Kravecovi a Broni Brejovej za ich pomoc, podporu, cenné rady a pripomienky. Tiež by som sa chcel poďakovať prof. Jozefovi Nosekovi za poskytnutie nepublikovaných dát, na ktorých sme otestovali našu metódu.

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím citovaných zdrojov.

.....

Abstrakt

Autor:	Martin Bobák
Názov práce:	Identifikácia variantov génov z dát sekvenovania novej generácie
Škola:	Univerzita Komenského v Bratislave
Fakulta:	Fakulta matematiky, fyziky a informatiky
Katedra:	Katedra informatiky
Vedúci práce:	Mgr. Tomáš Vinař, PhD.
Konzultant:	Mgr. Martin Kravec

V genómoch sa pomerne často vyskytujú varianty toho istého génu, ktoré je v rámci dlhej sekvencie DNA ťažké identifikovať. V práci sa snažíme identifikovať varianty génov z dát sekvenovania novej generácie. Problém najprv formalizujeme ako biologický problém, ktorý následne pretransformujeme na informatický problém. Informatický problém charakterizujeme pomocou teórie grafov. Zo spárovaných segmentov a referenčnej sekvencie génu konštruujeme graf variantov. Tento graf redukujeme pomocou kubického algoritmu na redukovaný graf variantov. Na týchto dvoch grafoch riešime problém hľadania najmenšieho počtu ciest, ktoré popisujú každú hranu. Na to používame celočíselný lineárny program. V závere práce prezentujeme dosiahnuté výsledky na experimentálnych dátach pochádzajúcich z kvasinky *Magnusiomyces magnusii*.

Kľúčové slová: sekvenovanie genómov, zostavovanie genómov, celočíselné lineárne programovanie

Abstract

Author:	Martin Bobák
Title:	Identification of gene variants from next-generation sequencing data
University:	Comenius University in Bratislava
Faculty:	Faculty of Mathematics, Physics and Informatics
Department:	Department of Computer Science
Supervisor:	Mgr. Tomáš Vinař, PhD.
Consultant:	Mgr. Martin Kravec

The variants of the same gene are a relatively common occurrence in the genome, which is within the long DNA sequence difficult to identify. In this thesis we try to identify gene variants from next-generation sequencing data. First we formalize the problem as a biological problem, which is subsequently transformed into an informatics problem. We characterize informatics problem using graph theory. We construct a graph of variants from paired end reads and the reference sequence of the gene. This graph is reduced using a cubic algorithm to reduced graph of variants. On these two graphs we solve problem of finding the smallest number of ways witch describe each edge. We use integer linear program to solve these problem. In the last chapter we present the results obtained on experimental data derived from yeast *Magnusiomyces magnusii*.

Keywords: genome sequencing, genome assemby, integer linear programming

Obsah

Úvod

Každý živý organizmus má genetickú informáciu uloženú v molekule DNA. Tú si môžeme predstaviť ako dlhý reťazec nad abecedou {A, C, G, T}. Jednotlivé písmenká v tomto reťazci reprezentujú štyri nukleotidy: adenín, cytozín, guanín a tymín. Pričom základnou jednotkou genetickej informácie je gén. Na gén sa dá pozrieť aj ako na kompletnú sekvenciu DNA, ktorá je potrebná na syntézu určitého produktu. Reťazec, ktorý reprezentuje DNA, poprípade gén, sa zisťuje pomocou sekvenovacích technológií. V súčasnosti sa však nedá sekvenovať ľubovoľne dlhá sekvencia. Namiesto toho sa sekvenujú len kratšie úseky, z ktorých sa neskôr poskladá daná vzorka. Zostavovanie genómov (súborov všetkých génov) nie je priamočiary proces. Jednými z pôvodcov nelineárnosti zostavovania genómov sú opakujúce sa úseky, ktoré sa nedajú jednoznačne zaradiť do hľadaného reťazca.

V genómoch sa pomerne často stáva, že nejaký gén sa skopíruje na iné miesto v rámci genómu. Následne prebiehajú mutácie (substitúcie, inzercie, alebo delécie) v oboch kópiách a kópie sa začnú mierne líšiť. Takýmto rôznym verziám pôvodného génu hovoríme varianty, ktoré sa dajú v rámci dlhej sekvencie DNA ťažko identifikovať. Mohlo by sa zdať, že stačí len zostaviť genóm, v ktorom by sa následne našli varianty skúmaného génu. Avšak sú to práve varianty, ktoré spôsobujú, že zostavovanie genómov je náročné. Zapríčinené je to tým, že varianty sú si navzájom veľmi podobné. Z tohto dôvodu algoritmy pracujúce s dátami, pochádzajúcimi zo sekvenovania novej generácie, majú s identifikáciou variantov génov značné problémy. Je ťažké v uvedených prípadoch rozlíšiť, či ide o sekvenovaciu chybu a takisto je v týchto dátach ťažké zaradiť jednotlivé krátke segmenty na správne miesto v rámci genómu. V práci navrhujeme metódu zostavujúcu jednotlivé varianty génov priamo zo sekvenovaných krátkych úsekov (segmentov), pochádzajúcich z rôznych variantov toho istého génu.

Prv než pristúpime k riešeniu úlohy, uvedieme v prvej kapitole čitateľa do kontextu sekvenovania a zostavovania DNA. V závere prvej kapitoly oboznámime čitateľa s nami navrhovanou formuláciou problému identifikácie variantov genómov z dát sekvenovania novej generácie, ako biologického problému (pozri problém 1). K tomuto problému, zostavovaniu variantov génu, následne navrhneme ekvivalentný informatický problém (pozri problém 2), ktorý je popísaný v druhej kapitole. Na vyriešenie informatického problému sme navrhli dátovú štruktúru tzv. *grafu variantov*. V ďalšej, tretej kapitole uvedený graf zredukujeme na redukovaný graf variantov. Algoritmus na hľadanie redukovaného grafu variantov má časovú zložitosť $O(n^3)$. V predposlednej kapitole pristupujeme k samotnému riešeniu informatického problému, definovaného v predchádzajúcej kapitole. Keďže sa jedná o NP-ťažký problém, riešime ho pomocou celočíselného lineárneho programovania. V poslednej kapitole otestujeme náš prístup na kvasinkových dátach, ktoré vznikli pomocou sekvenovacej technológie druhej generácie Illumina[1].

Hlavné prínosy diplomovej práce sú nasledovné:

- definovanie zostavovania variantov ako biologický problém
- transformácia biologického problému na informatický problém ako problém hľadania ciest v grafe variantov
- definovanie grafu variantov
- definovanie redukovaného grafu variantov
- navrhnutie celočíselného lineárneho programu na hľadanie ciest v grafe variantov a jeho následná modifikácia pre redukovaný graf variantov

[1]Konkrétne pracujeme so spárovanými segmentami.

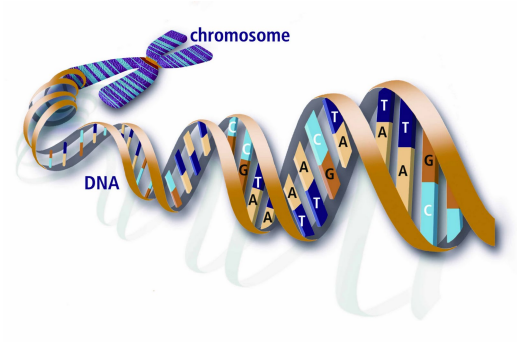
Kapitola 1

Sekvenovanie a zostavovanie genómov

V tejto kapitole sa oboznámime so sekvenovaním a zostavovaním genómov. Najprv uvedieme základnú biologickú terminológiu, ktorú budeme používať v rámci celej práce. Potom prejdeme k samotnému sekvenovaniu DNA. Vysvetlíme si rozdiely medzi starou a novou generáciou sekvenovania. Vzhľadom na to, že nie je možné sekvenovať ľubovoľne dlhé sekvencie, sekvenovacie technológie produkujú sekvencie úsekov DNA – segmenty. Až z osekvenovaných segmentov DNA sa zostaví jej celková sekvencia. Na tieto účely slúžia metódy na zostavovanie genómov. V závere kapitoly zoznámime čitateľa s našou formalizáciou identifikácie variantov génov ako biologického problému, s ktorým sa budeme zaoberať v ďalších kapitolách.

1.1 Základné pojmy z genetiky

Deoxyribonukleová kyselina (**DNA**) je polymér, ktorý sa skladá z nukleotidov. Jednotlivé nukleotidy v dvojrozmernom priestore tvoria dve polynukleotidové vlákna spletené do dvojzávitnice (pozri obrázok 1.1). Nukleotidy sa medzi sebou líšia dusíkatými bázami. V DNA sa vyskytujú štyri dusíkaté bázy. Dve bázy sú odvodené od purínu: adenín a guanín. Dve bázy s odvodené od pyrimidínu: cytozín a tymín. Medzi vlák-



Obr. 1.1: Štruktúra molekuly DNA. [Hal06]

nami DNA platí komplementarita dusíkatých báz. To znamená, že adenín tvorí pár s tynímom (A – T) a guanín s cytozínom (G – C).

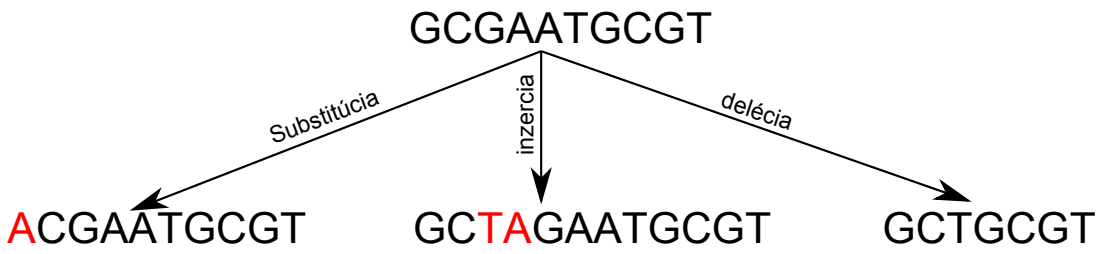
DNA plní v bunke funkciu nositeľky genetickej informácie. Tá je v nej zaznamenaná práve sledom uvedených štyroch dusíkatých báz označených písmenami A, C, G, T. Ich poradie hrá dôležitú úlohu pri prenose genetickej informácie. Molekulu DNA reprezentujeme dlhým reťazcom nad abecedou $\Sigma = \{A, C, G, T\}$.

Funkčné úseky, ktoré sú ďalej prepísané do RNA sa nazývajú **gény**. V génoch je uložená informácia určená na syntézu určitého produktu. Najčastejšie je to bielkovina. Súbor všetkých génov nachádzajúcich sa v bunke sa nazýva **genóm**.

V genóme je častým javom, že sa niektoré gény skopírujú na iné miesta. Na týchto miestach môžu podliehať rôznym génovým mutáciám, čím sa začnú vyvíjať nezávisle od svojej predlohy. **Génové mutácie** často vytvárajú len malé zmeny genetickej informácie. Účinok takýchto zmien na gén je však rôzny. Zmutovaný gén si totiž môže zachovať plnú funkčnosť oproti pôvodnému génu, ale takisto môže jediná mutácia viesť až k úplnému znefunkčneniu proteínu, ktorý je mutovaným génom kódovaný.

Z hľadiska vplyvu mutácie na DNA rozpoznávame tieto typy mutácií
(pozri obrázok 1.2):

- **substitúcie** - zámeny nukleotidu za iný nukleotid, ktoré rozlišujeme na:
  - **tranzície** - vymení sa purín za purín alebo pyrimidín za pyrimidín (napríklad z A na G)
  - **transverzie** - vymení sa purín za pyrimidín alebo opačne (napríklad z A na T)
- **inzercie** - vloženie jedného alebo viacerých nukleotidov
- **delécie** - strata jedného alebo viacerých nukleotidov



Obr. 1.2: Ukážka substitúcie, inzercie a delécie na sekvencii GCGAATGCGT

Génovou mutáciou vznikajú rôzne verzie pôvodného génu, ktorým hovoríme **varianty génu**.

1.2 Sekvenovanie DNA

Sekvenovanie DNA je proces v ktorom sa zisťuje presné poradie dusíkatých báz vo vzorke (určujeme primárnu štruktúru DNA). DNA sa sekvenuje z mnohých dôvodov. Jednak preto, že zo sekvencie DNA dokážeme určiť sekvencie RNA a poradie aminokyselín v proteínoch, ktoré vznikli zo skúmaného úseku DNA (poradie báz slúži ako predloha na vybudovanie konkrétnej RNA, či proteínu skladajúceho sa z aminokyselín). Ale aj preto, že sekvenovať DNA je oveľa jednoduchšie a lacnejšie, ako sekvenovanie RNA, či proteínov.

V súčasnosti existuje viacero typov sekvenovacích technológií. Jednou z najrozšírenejších metód na sekvenovanie DNA bola donedávna Sangerova metóda. Hlavnou nevýhodou tejto metódy sú však vysoké náklady na jej prevádzkovanie a pomalá rýchlosť produkcie dát. Tieto nedostatky sa usilujú odstrániť nové sekvenovacie technológie [WGF10]. Prechod zo Sangerovej metódy na nové technológie nie je však priamočiary. Je to spôsobené hlavne zmenou povahy samotných dát. Preto keď chceme spracovať dáta z nových sekvenovacích technológií, musíme pre ne nevyhnutne vyvinúť aj nové algoritmy, ktoré ich spracujú a vyhodnotia.

1.3 Sekvenovacie technológie

Sangerovo sekvenovanie [SF77] je hlavným reprezentantom sekvenovacích technológií prvej generácie. Jednou z najväčších nevýhod Sangerovho sekvenovania je, že sa ním nedajú sekvenovať ľubovoľne dlhé vzorky DNA. Týmto spôsobom je možné spoľahlivo sekvenovať len 500 – 1000 báz. Ďalším dôležitým faktorom v istom zmysle proti používaniu tejto metódy, je spoľahlivosť výsledku. Nie vždy je totiž možné jednoznačne určiť zo sekvenovacieho profilu, aký nukleotid sa nachádza na danej pozícií. Tento problém však vieme sčasti riešiť pomocou relatívne jednoduchých pravdepodobnostných skórovaní kvality výsledku, čo však nemení nič na tom, že dlhšie sekvencie nevieme spracovať.

Rozdiel medzi novými technológiami a Sangerovou metódou nie je len v celkovom náraste produkcie segmentov, ale aj v ich kvalite. Segmenty, ktoré produkujú nové technológie sú podstatne kratšie (ich dĺžka je približne 100 báz). Tým je zostavovanie genómu omnoho náročnejšie, pretože sekvenovanie sa stáva omnoho viac nejednoznačné. Práve táto skutočnosť spôsobuje, že výsledok je často fragmentárny. Čiže problematika sekvenovania genómu sa akoby posunula k aplikáciám, ktoré zostavujú genóm. Na druhej strane však paralelne spracovávajú obrovské množstvo dát, čím sa stávajú rýchlejšie, ale aj lacnejšie než Sangerovo sekvenovanie. Nové technológie produkujú často segmenty v pároch. To znamená, že segmenty sú spárované do dvojíc. V každej dvojici poznáme ich poradie, orientáciu a približnú vzdialenosť vzhľadom na pôvodnú DNA sekvenciu (napríklad sekvenovacia technológia Illumina produkuje segmenty, ktoré sú dlhé približne 100 báz a vzdialenosť medzi nimi je približne 60 báz). Príkladom takýchto technológií sú Illumina [WGF10] a AB SOLID [CJB10].

Keď to zhrnieme, dáta pochádzajúce z novej generácie sekvenovania sa vyznačujú týmito základnými vlastnosťami:

- pochádzajú z obrovského množstva segmentov sekvenovaného paralelne z tej istej vzorky
- sekvenovanie týmto spôsobom je rýchlejšie a lacnejšie
- segmenty sú kratšie

V tejto práci pracujeme s dátami pochádzajúcimi z druhej generácie sekvenovacích technológii. Z porovnania dĺžok segmentov pochádzajúcich z prvej generácie sekvenovacích technológií voči dĺžke segmentov pochádzajúcich z novej generácie sekvenovacích technológií je zrejmé, že zostavovanie genómov pomocou segmentov z novej generácie sekvenovacích technológií je oveľa náročnejšie než zostavovanie genómov pomocou segmentov z prvej generácie sekvenovacích technológií. Gény s ktorými pracujeme pochádzajú z kvasiniek. Ich dĺžka sa pohybuje približne od 1000 do 1500 báz. Pri Sangerovom sekvenovaní bolo pomerne ľahké určiť sekvencie takýchto génov. Dokázali sme ich totiž pokryť jedným až dvoma takýmito segmentami. Na segmenty pochádzajúce z novej generácie sekvenovacích technológii sa však takýto prístup už nedá použiť.

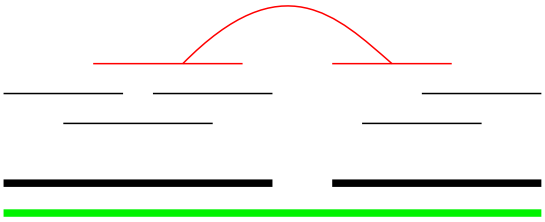
1.4 Zostavovanie genómov

V súčasnosti ešte nejestvuje technológia, ktorá dokáže sekvenovať celý genóm. Výsledok zo súčasných sekvenovacích technológií je v podobe mnohých segmentov, ktoré sa snažíme poskladať do súvislého reťazca - genómu. Zostavovanie genómov nie je však priamočiary proces. Vyskytujú sa v ňom nasledovné problémy.

Počas sekvenovania môžu vznikať *chyby*, teda dostávame čiastočne odlišné výsledky. Celkovo je veľkým problémom rozlíšiť, o aký typ chyby ide (chyba, ktorá vznikla v konkrétnom experimente, alebo chybnou vzorkou). Vzorka môže byť napríklad znečistená, poškodená iným genetickým materiálom a podobne. Ďalším problémom je polymorfizmus vo vzorke. Ako príklad si vezmime diploidný organizmus (napr. človeka). Tieto organizmy majú dve kópie každého chromozómu. Jednu získali od otca a druhú od matky. Takže je celkom možné, že budú v nich existovať pozície, na ktorých budú rôzne bázy. Keď však sekvenujeme, tak tento jav nedokážeme zohľadniť. Spracúvame totiž segmenty, ktoré sú zmesou oboch chromozómov.

Ďalším problémom, s ktorým zápasia sekvenovacie technológie, je *nerovnomerné pokrytie* genómu segmentmi. Dokonca je možné, že niektoré miesta nebudú vôbec pokryté. Tieto sa nám však nepodarí rekonštruovať, čo spôsobí, že dostaneme fragmentárny výsledok.

Posledným problémom sú *repetatívne sekvencie*. Genómy, ktoré obsahujú tieto sekvencie je veľmi ťažké zrekonštruovať. Správne zostaviť genóm nám pomáhajú spárované segmenty (pozri obrázok 1.3). Sú to dvojice segmentov so známou vzdialenosťou. Vznikajú napríklad tým spôsobom, že sa sekvenuje úsek z oboch strán. Keď sú segmenty dostatočne vzdialené, dokážu zjednoznačniť úseky s repetitívnymi sekvenciami.



Obr. 1.3: Segmenty (znázornené tenkou čiernou a červenou čiarou) sú zarovnané k referenčnej sekvencii génu (znázornená zelenou čiarou). Spárované segmenty (znázornené červenou farbou) pomáhajúce určiť poradie jednotlivých pospájaných častí v genóme (znázornené hrubou čiernou čiarou).

Krátke segmenty (napr. tie, ktoré produkujú nové sekvenovacie technológie) taktiež sťažujú zostavenie genómu. Ďalšími parametrami, ktoré ovplyvňujú zostavovanie, sú: dĺžka zostavovanej sekvencie a počet segmentov pokrývajúcich danú pozíciu vo vzorke - pokrytie. Vstupom do tejto úlohy je množstvo krátkych prekrývajúcich sa podreťazcov a naším cieľom je zostaviť pôvodný reťazec.

Jedným z najväčších problémov, ktoré brzdia nástup nových sekvenovacích technológií je, že programy zostavujúce genómy pomocou dát zo Sangerovho sekvenovania sa nedajú priamo použiť na dáta pochádzajúce z nových sekvenovacích technológií. Spoliehajú sa totiž na dlhé segmenty, pričom tieto technológie majú svoje špecifické chyby. Okrem toho nemajú ani dostatočný výkon, aby mohli spracovať množstvo segmentov vygenerovaných týmito technológiami.

Najprirodzenejšou formuláciou tejto úlohy je **problém najkratšieho spoločného nadslova**. V uvedenom probléme máme daných niekoľko reťazcov. Naším cieľom je nájsť najkratší reťazec, ktorý však obsahuje všetky reťazce (tie, čo sme dostali na vstupe), ako súvislé podreťazce. V riešení nám ide o maximálne využitie prekryvov medzi reťazcami zo vstupu. Z pohľadu teoretickej informatiky ide

o NP-ťažký problém (t.j. nepoznáme efektívny spôsob ako ho optimálne riešiť) [GMS80]. Nevýhodou tohoto prístupu však je, že neberie do úvahy rôzne javy, ktoré sú súčasťou sekvenovania:

- sekvenovacie chyby
- polymofizmus vo vzorke
- kontaminácia cudzím genetickým materiálom
- nerovnomerné pokratie vzorky segmentami
- repetetívne sekvencie
- existujú genómy, keď výsledkom nie je najkratšie spoločné nadslovo

Ako vidíme sformulovať problém pre zostavovanie genómov tak, aby v sebe zahŕňal všetky rysy dát pochádzajúcich zo sekvenovacích technológií, nie je vôbec ľahká úloha. Preto výskumníci, zaoberajúci sa touto problematikou, sa snažia nájsť pre zostavovanie genómov rôzne heuristiky.

1.4.1 De novo zostavovanie genómov

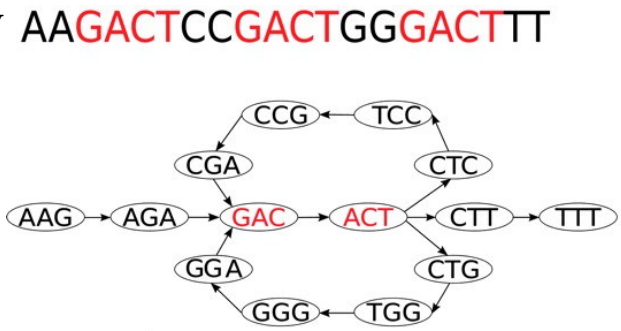
V praxi sa na zostavovanie genómov pozeráme, akoby sme chceli riešiť veľké puzzle bez toho, aby sme vopred poznali výsledný obrázok. Opakujúce sa časti sekvencie zodpovedajú podobne zafarbeným kúskom (napríklad oblohe). Ľahko sa dá rozpoznať, že ide o NP-ťažký problém. Primárnym zdrojom takéhoto prístupu sú opakujúce sa časti, ktoré spôsobujú nejednoznačnosť riešenia (najmä tie, ktoré sú dlhšie ako segment). V takýchto situáciách program danú časť háda. Tento prístup však vedie s veľkou pravdepodobnosťou k chybám. Druhou možnosťou je obmedziť sa na neopakujúce sa časti genómu. Týmto spôsobom však budeme produkovať fragmentárne dáta. Tento problém nám čiastočne pomôžu riešiť spárované fragmenty.

Na to, aby sme získali výsledky porovnateľné so Sangerovým sekvenovaním, potrebujeme 25 – 30 násobné pokrytie genómu (keďže naviac nemáme žiadne informácie, akými sú napríklad spárované fragmenty) [KDB$^+$06].

Problém sa však dá napríklad riešiť nasledovným „greedy“ prístupom. Najprv si vyberieme špeciálne segmenty – „semienka“. Každé semienko predlžujeme segmentami, ktoré sa dostatočne prekrývajú so semienkami. Postup opakujeme dovtedy, až

pokiaľ nie je jednoznačný. Vďaka tomuto prístupu robíme podstatne menej chýb spôsobených opakujúcimi sa segmentami. Na druhej strane sa nám podarí semienka predĺžiť len o málo báz. Vysoká miera fragmentárnosti je spôsobená hlavne krátkymi segmentami. Časovo je to však veľmi náročný prístup. Zapríčiňuje to najmä množstvo dát, ktoré produkujú nové sekvenovacie technológie.

Väčšina súčasných technológií využíva prevažne **deBruijnove grafy** [PTW01] (v ďalšom texte popíšeme prispôsobenie tohto prístupu na dáta obsahujúce krátke segmenty). Na začiatku sa množina segmentov rozstrihá na kratšie segmenty (podreťazce) s rovnakou dĺžkou. Vrcholy v grafe sú tieto podreťazce. Hrany sú medzi tými vrcholmi, ktoré boli susedmi v pôvodnom segmente.



Obr. 1.4: DeBruijnov graf. [CBP09]

Genóm zostavíme tak, že prejdeme po každej hrane práve raz (hľadáme Eulerovský ťah). Hoci vo výsledku je mierna fragmentárnosť, možno ho považovať za pomerne kvalitný. Príkladom takejto technológie je Velvet [ZB08].

Veta 1. *Orientovaný graf má Eulerovský ťah z vrcholu* u *do vrcholu* v *práve vtedy, keď po pridaní hrany* (u,v) *je graf silne súvislý a v každom vrchole sa počet vchádzajúcich hrán rovná počtu vychádzajúcich hrán.*

Ak je táto podmienka splnená, Eulerovský ťah dokážeme hľadať v grafe G s časom $O(n + m)$, kde n je počet vrcholov grafu G a m je počet hrán grafu G. Na deBruijnových grafoch, zostavených z reálnych dát, sa často stáva, že buď v nich neexistuje eulerovský ťah, alebo v nich existuje viacero Eulerovských ťahov. Preto sa tento prístup dopĺňa rôznym množstvom heuristík, ktoré umožňujú:

- detegovať sekvenovacie chyby (v takomto prípade sa vrcholy a hrany zodpovedajúce týmto chybám odstránia z grafu)
- detegovať repetetívne úseky (hrany zodpovedajúce týmto úsekom sa znásobia)
- „zliepať vrcholy“ do jedného spoločného vrchola (fixovanie si určitého úseku)

- využívanie spárovaných segmentov

V súčasnosti prebieha výskum, ktorého cieľom je zapojiť do deBruijnovho grafu spárované segmenty. Spárované segmenty majú potenciál pomôcť pri zostavovaní genómov. Preto sa často používajú v záverečnej fáze na zlepšenie celkového výsledku. Ambíciou týchto nových projektov je zapojiť spárované segmenty priamo do procesu tvorby deBruijnovho grafu a nie ich použitie v záverečnej fáze. Príkladmi takýchto projektov sú: spárovaný deBruijnov graf [MPC+11], SPAdes [BNA+12] a Pathset graph [PAS+13].

Dôležitým parametrom heuristík využívajúcich deBruijnove grafy je k, ktoré predstavuje dĺžku podreťazcov vo vrcholoch deBruijnovho grafu. Ak si zvolíme veľmi malé k, potom budeme mať v grafe veľký počet nejednoznačností zapríčinenými opakujúcimi sa *k-ticami*. Naopak, ak budeme pracovať s veľmi veľkým k, potom pôjde o veľmi náročné výpočty (bude zaťažená najmä operačná pamäť počítača). Treba si však uvedomiť, že veľké k nemusí pritom jednoznačne vyriešiť všetky problémy.

Momentálny úspech deBruijnových grafov tkvie najmä v tom, že sú použiteľné na dáta pochádzajúce z novej generácie sekvenovacích technológií, produkujúcich veľké množstvo krátkych segmentov. Pri nich si totiž môžeme zvoliť menšie k a nestratiť pritom veľké množstvo informácií.

1.4.2 Resekvenovanie

Ako sme uviedli v predchádzajúcich častiach, v zostavovanej DNA sa vyskytujú aj nepokryté úseky. Takto v celkovom výsledku vznikajú medzery, o ktorých nevieme nič povedať. Ďalším problémom, s ktorým sa stretávame pri zostavovaní DNA sekvencie, sú opakujúce sa úseky. Respektíve úseky, ktoré sa sú si príliš podobné. Tieto úseky však nedokážeme jednoznačne rozlíšiť pomocou segmentov. Predovšetkým tieto dva javy spôsobujú, že výsledný genóm bude fragmentárny. Fragmentárnosti sa žiaľ momentálne dá ťažko vyhnúť. Jedným z riešením je resekvenovanie, ktoré môže pomôcť pri nejasnostiach, či pri zisťovaní sekvencie medzier. Keďže cena sekvenovania klesá, začína sa v súčasnosti stále viac uvažovať aj o možnostiach resekvenovania. Je to jeden z ďalších spôsobov, ako sekvenovať celý genóm. Metódy, ktoré boli doposiaľ prezentované však nič nepredpokladali o skúmanom genóme. Resekvenovanie totiž predpokladá, že už čiastočne poznáme genóm, o ktorý sa zaujímame. Skúmané dáta sa v tomto prípade snažíme zarovnať na sekvenovaný známy genóm. Keďže

nové sekvenovacie technológie produkujú veľké množstvo dát, súčasťou resekvenovacích technológií sa stali metódy umožňujúce rýchle prehľadávanie textu. Takouto metódou je napríklad FM index [FM01]. Metódy, akými sú FM index, pomáhajú vyhľadávať segmenty, ktoré sa dokonale, respektíve postačujúco, zhodujú so sekvenciou známeho genómu (ku ktorému chceme jednotlivé segmenty zarovnať). Tento prístup nám umožňuje nielen zlepšovať kvalitu už existujúcich sekvencií, ale resekvenovaním môžeme sledovať aj rôznorodosť a podobnosť v rámci jedného druhu. Znovu sa dostávame k otázke dĺžky segmentu. Čím je však segment dlhší, tým presnejšie je zarovnanie na známy genóm. Ak chceme spracovať dáta z nových technológií, musíme predovšetkým dokázať spracovať veľké množstvo dát. Na druhej strane nám tieto technológie umožňujú resekvenovať celý genóm. V súčasnosti sa pracuje na efektívnych zarovnávacích algoritmoch a na dokonalejšom určovaní chýb v dátach skladajúcich sa z krátkych segmentov (na vypracovaní chybových vzorov). Príkladom takéhoto prístupu je MAQ [LRD08], SOAP [LLK08], ZOOM [LZZ+08] a Bowtie [LTPS09].

1.5 Cielené sekvenovanie

Metóda, ktorá chce spracovať a analyzovať dáta pochádzajúce zo sekvenovania novej generácie, musí dokázať spracovať obrovské množstvo dát. V jednom behu sekvenovania dokážu tieto technológie vyprodukovať až 50 GB dát [MDH11]. V dôsledku tohoto sú výskumníci, pri spracúvaní a analýze, často obmedzovaný časovými a pamäťovými požiadavkami uvedených technológií. Preto sa v niektorých prípadoch namiesto celého genómu skúma len jeho určitá časť. Príkladmi technológií cieleného sekvenovania a zostavovania sú: TASR [WH11] a Mapsembler [PC12]. Gk polia [PSL+11] sú príkladom dátovej štruktúry umožňujúcej dotazy nad veľkým množstvom segmentov. V článku [ZRS+12] autori prezentujú pravdepodobnostný model na aproximáciu distribúcie výskytov slov v sekvenovacích dátach (bez použitia technológií na zostavovanie či zarovnávanie dát). Ďalším príkladom lokálneho spracovania dát zo sekvenovania novej generácie je GapFiller [NVP12]. GapFiller používa spárované segmenty na lokálne doplnenie sekvencií medzier.

1.6 Identifikácia variantov génov ako biologický problém

V genómoch (súbore všetkých génov) sa pomerne často vyskytuje, že určitý gén sa skopíruje na iné miesto v rámci genómu. Následne prebiehajú mutácie (substitúcie, inzercie, alebo delécie) v oboch kópiách a následne sa kópie začnú mierne líšiť. Takýmto verziám pôvodného génu hovoríme varianty.

Náš prístup, popísaný v tejto práci, je pipelinová metóda cieleného sekvenovania. To čo nás zaujíma sú len varianty určitého génu. Zvyškom genómu sa nezaoberáme. Varianty génov, najmä tie, čo sú si veľmi podobné, je v rámci dlhej sekvencie DNA ťažké identifikovať. Je to spôsobené predovšetkým tým, že pri určovaní variantov, ktoré sú si veľmi podobné, je nevyhnutné odlíšiť segmenty, ktoré pochádzajú z daných variantov od segmentov, ktoré vznikli ako sekvenovacie chyby. Okrem toho treba vziať do úvahy, že tieto varianty sa budú správať ako repetetívne úseky, čo je všeobecný problém, ktorý sťažuje zostavovanie genómov. V práci sa snažíme varianty génov identifikovať priamo zo sekvenovaných krátkych úsekov (segmentov), pochádzajúcich z rôznych variantov toho istého génu. Ako prvý krok je potrebné formalizovať problém zostavovania variantov génu. Navrhujeme nasledovnú formalizáciou tohoto problému.

Problém 1. ***Zostavovanie variantov génu***
Vstup: *osekvenované krátke segmenty DNA a referenčná sekvencia génu*
Problém: *zostaviť z osekvenovaných segmentov DNA varianty referenčného génu*
Výstup: *poskladané varianty génu*

Môže sa zdať, že najjednoduchšie riešenie uvedeného problému získame tým spôsobom, že najskôr zostavíme celý genóm, a následne nájdeme varianty vybraného génu. Práve prítomnosť variantov, ktoré sú si navzájom veľmi podobné, však predstavuje značný problém pre algoritmy pracujúce s dátami pochádzajúcich zo sekvenovania novej generácie. Je totiž veľmi ťažké určiť, či v danej alternatíve ide o chybu sekvenovania, alebo daný gén má viacej variantov. V podstate sa tu stretávajú dva základné problémy, ktoré robia zostavovanie genómov náročným: opakujúce sa úseky a sekvenovacie chyby. V takejto situácii pomôže, ak sú jednotlivé segmenty spárované. To znamená, že máme k dispozícii dvojice segmentov, ktoré sú vzdialené niekoľko desiatok báz (pozri Obr. 1.3).

Kapitola 2

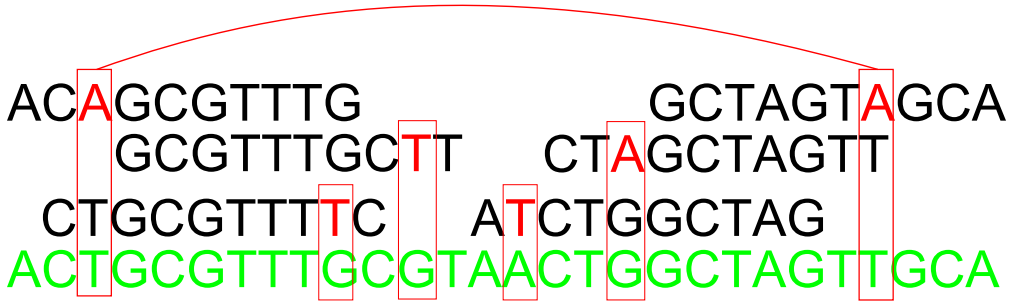
Graf variantov

V tejto kapitole zoznámime čitateľa s našou transformáciou biologického problému (Problém 1) na problém informatický. Pôjde o hľadanie ciest v tzv. grafe variantov. *Graf variantov* je nami navrhovaný grafový model, ktorý reprezentuje závislosti medzi jednotlivými alternatívami. Každý variant génu je v ňom reprezentovaný cestou. Čím sa úloha identifikácie variantov génu zmení na hľadanie týchto ciest. To však nestačí na reálne nasadenie tejto metódy. Na skutočných dátach často totiž vznikajú grafy s veľkým počtom hrán, ktoré sa však nevyskytujú vo výsledných variantoch (sú len akýmsi obmedzením, ktoré musia spĺňať hľadané varianty). Z tohto dôvodu sa v nasledujúcej kapitole pokúsime tento graf zredukovať. Vďaka tejto redukcii sa táto metóda stane použiteľná aj pre praktické účely.

2.1 Graf variantov

Predtým, ako prejdeme k samotnej transformácii biologického problému na informatický, si definujeme graf variantov, pomocou ktorého spravíme uvedenú transformáciu.

Na začiatku sekvenované segmenty zarovnáme k referenčnému génu, napríklad pomocou algoritmu Blat [Ken02], prípadne Tophat [TPS09]. To znamená, že pre každý segment určíme, z ktorej časti referenčného génu pochádza. Z každého zarovnaného segmentu môžeme určiť pozície, v ktorých sa líši oproti referenčnému génu. Jednotlivé bázy, odlišné od referenčného génu, nazveme *alternatívy* a pozíciu, na ktorej sa alternatívy vyskytujú, označíme ako *nejednoznačnú* (pozri obrázok 2.1.).



Obr. 2.1: Ukážka hľadania alternatívnych báz na nejednoznačných pozíciách. Šesť segmentov (čierne reťazce) je zarovnaných k referenčnému génu (zelený reťazec), z ktorých dva sú spárované (znázornené červenou čiarou). Určenie alternatív znamená nájsť bázy variantov, ktoré sa líšia od báz referenčného génu. Na obrázku sú to bázy znázornené červenou farbou. Určenie nejednoznačných pozícií znamená stanovenie pozícií vzhľadom na referenčný gén, na ktorých sa nachádzajú alternatívne bázy, t.j. v tomto prípade sú to pozície 3, 10, 12, 15, 18 a 25.

Definícia 1. Alternatíva *je báza nachádzajúca sa vo variante génu, ktorou sa odlišuje variant génu od referenčného génu.*

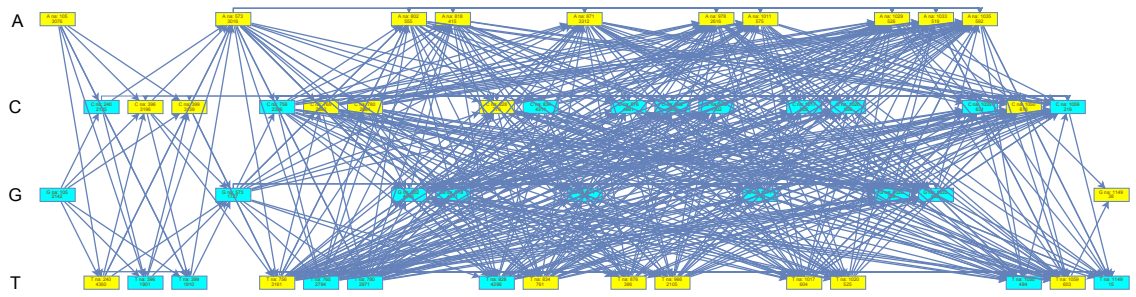
Definícia 2. Nejednoznačná pozícia *je pozícia, na ktorej sa odlišuje variant génu od referenčného génu.*

Aby sme reprezentovali závislosti medzi jednotlivými alternatívami, zostavíme tzv. *graf variantov*. Graf variantov vyjadruje, ktoré zmeny sa nachádzajú v jednom segmente, či páre segmentov, a teda aj v jednej variante génu.

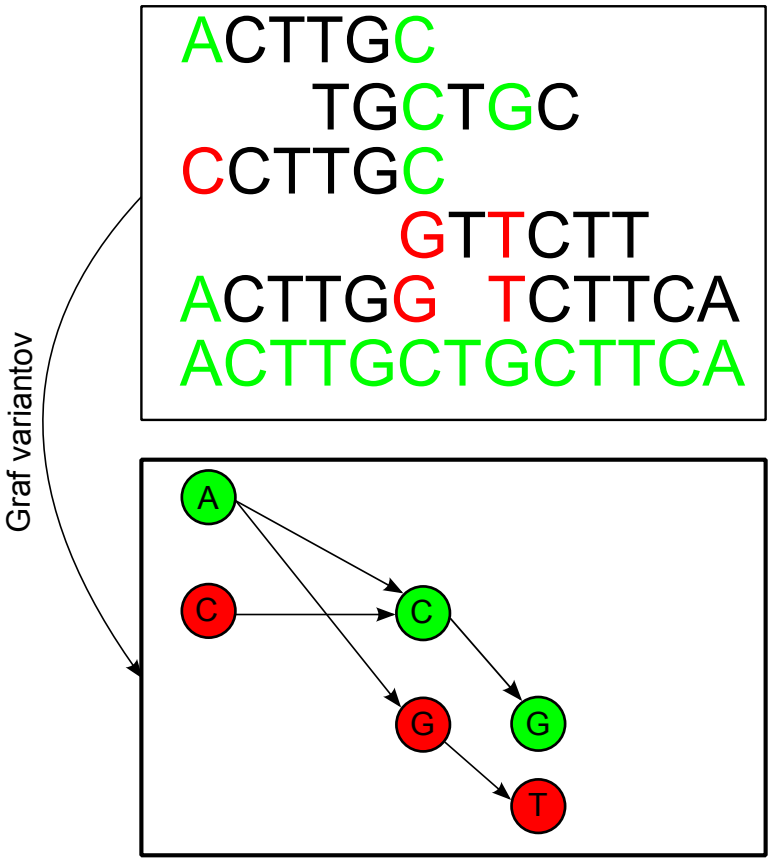
Definícia 3. Graf variantov *je orientovaný acyklický graf, ktorý pre každú nejednoznačnú bázu referenčnej sekvencie obsahuje niekoľko vrcholov reprezentujúcich všetky alternatívy na danej pozícií. Hrana spája dva vrcholy* (u, v)*, ak existujú spárované segmenty,* [2] *alebo segment, ktorý naraz obsahuje obe alternatívy zodpovedajúce vrcholom* u *a* v*. Orientácia hrán je v smere referenčnej sekvencie.*

Príklad grafu variantov môžeme vidieť na obrázku 2.2.

[2]Ide o explicitne spárovanú dvojicu segmentov, ktorú sme dostali ako výstup zo sekvenátora.



Obr. 2.2: Graf variantov z experimentálnych dát kvasinky *Magnusiomyces magnusii*. Vrcholy sú horizontálne usporiadané podľa pozície v referenčnej sekvencií a predstavujú štyri bázy. Vertikálne sú vrcholy usporiadané podľa pozície. Modré vrcholy pochádzajú z variantov, žlté vrcholy pochádzajú z referenčného génu.



Obr. 2.3: Vznik grafu variantov zo zarovnania. V hornej časti sa nachádza šesť segmentov, pod nimi je sekvencia referenčného génu (znázornená zelenou farbou). Rozlišujeme dva typy vrcholov: alternatívy (červená farba) a vrcholy zodpovedajúce referenčnému génu (zelená farba).

Algoritmus na zostavovanie grafu variantov na základe osekvenovaných krátkych segmentoch DNA, sekvencii referenčného génu a ich zarovnania k sekvenii referenčného génu, ktoré má na vstupe, vyzerá nasledovne. Na základe zarovnania sa identifikujú nejednoznačné pozície. Na týchto pozíciách sa určia alternatívy, ktoré budú predstavovať vrcholy grafu variantov. Nakoniec sa pridajú do grafu variantov hrany zodpovedajúce vstupným dátam (pozri obrázok 2.3).

Veta 2. *Algoritmus na zostavovanie grafu variantov pre každý vstup vráti po konečnom počte krokov správny výsledok. Označme si počet zarovnaných segmentov k referenčnému génu n, maximálnu dĺžku segmentu d a maximálny počet nejednoznačných pozícií v segmente z. Časová zložitosť algoritmu na zostavovanie grafu variantov je polynomiálna, konkrétne $O(nd + nz^2)$.*

Dôkaz. Prvá časť vety, týkajúca sa korektnosti algoritmu, je zrejmá. Druhú časť vety, o časovej zložitosti nášho algoritmu, si ukážeme nasledovne. Identifikovať nejednoznačné pozície, na základe zarovnania, vieme s časovou zložitosťou $O(nd)$ – porovnáme každý segment s jemu zodpovedajúcou časťou referenčného génu. S rovnakou časovou zložitosťou vieme pridať vrcholy do grafu variantov. Hrany do grafu variantov pridáme s časovou zložitosťou nz^2 – hranu pridávame medzi všetky vrcholy pochádzajúce z jednej dvojice spárovaných segmentov. Čiže výsledná časová zložitosť bude $O(nd + nz^2)$. □

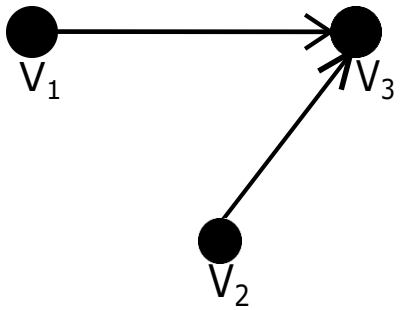
2.2 Identifikácia variantov génov ako informatický problém

Zatiaľ sme si zadefinovali graf variantov a navrhli sme algoritmus na jeho zostrojenie. Teraz si ukážeme, ako previesť biologický problém zostavovania variantov na informatický problém. Tento informatický problém bude využívať graf variantov.

V grafe variantov sú jednotlivé varianty génov vyjadrené cestami vedúcimi z počiatočného vrchola (vrchol do ktorého nevchádza žiadna hrana) do koncového vrchola (vrchol z ktorého nevychádza žiadna hrana). Nie všetky hrany musia byť zahrnuté v takýchto cestách. Každá hrana však musí byť cestou *popísaná*.

Definícia 4. *Cesta c* popisuje *hranu* (u, v) *nachádzajúcu sa v grafe G, ak sa dvojica vrcholov u, v nachádza na ceste c.*

Na tomto mieste sa vynára otázka, či cesta je korektná reprezentácia pre varianty génu. Môžu nastať situácie, kedy tomu tak nie je? Problém pri reprezentácií variantov génov, ako ciest, môžeme vidieť na obrázku 2.4. V tomto príklade existuje variant génu obsahujúci všetky tri nejednoznačné bázy/vrcholy v_1, v_2 aj v_3, ale neexistuje segment, ktorý by súčasne pokrýval vrchol v_1 a v_2.



Obr. 2.4: Príklad malého grafu variantov pre 3 vrcholy.

Formálne každá alternatíva pochádzajúca z jedného variantu musí spĺňať nasledujúce tvrdenie.

Veta 3. *Označme si d maximálnu dĺžku segmentu a v maximálnu vzdialenosť dvoch spárovaných segmentov. Pre všetky alternatívy pochádzajúce z jedného variantu génu, ktoré sa nachádzajú vo vzdialenosti $d+v$ existuje segment, alebo spárované segmenty, ktoré ich obsahujú.*

V našich experimentálnych dátach sa však takéto udalosti nevyskytujú, pretože spárované segmenty majú medzi sebou vhodnú vzdialenosť, veľkosť a dostatočné pokrytie referenčnej sekvencie. Preto v našom prípade je cesta dobrou reprezentáciou variantu génu.

Teraz nám už nič nebráni, aby sme formulovali hľadanie variantov génu, ako hľadanie minimálnej množiny ciest, v ktorej bude popísaná každá hrana grafu variantov.

Problém 2. ***Hľadanie ciest v grafe variantov***
Vstup: *graf variantov G*
Problém: *Nájsť k ciest na grafe G, ktoré popisujú všetky hrany v G, kde k je minimálne možné.*
Výstup: *k ciest na grafe G*

Veta 4. *Problém hľadania ciest v grafe variantov je NP-ťažký [KBBV13].*

Biologický problém 1 sme teda vyjadrili ako hľadanie minimálneho počtu ciest na grafe variantov, ktoré popisujú všetky jeho hrany (pozri obrázok 5.6). V ďalších kapitolách venujeme pozornosť riešeniu tohto informatického problému.

2.2.1 Riešiteľnosť

Keďže problém 2 je NP-ťažký, rozhodli sme sa ho riešiť pomocou celočíselného lineárneho programu.

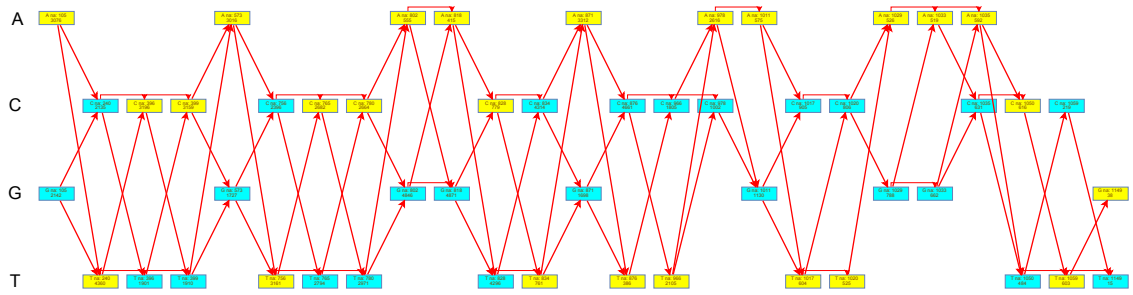
V tejto fáze je síce tento problém z teoretického hľadiska riešiteľný, no pre praktické účely je takéto riešenie nevyhovujúce. Celočíselný lineárny program musí zvažovať obrovský počet ciest, v dôsledku čoho narastá jeho časová zložitosť. Pritom to nie je nevyhnutné. Ako ukážeme v nasledujúcej kapitole, počet hrán grafu variantov sa dá totiž podstatne zredukovať, čím sa naša metóda stáva použiteľná aj pre praktické účely. Podrobnejšie sa tejto problematike venujeme v kapitole 4.

Kapitola 3

Redukovaný graf variantov

Predtým, než na reálnych dátach začneme hľadať cesty, ktoré popisujú všetky hrany v grafe variantov, určíme hrany, ktorými budú tieto cesty prechádzať. Takéto hrany neskôr využijeme na redukciu času pri riešení problému 2. Čiže namiesto toho, aby sme hľadali cesty v grafe variantov, budeme ich hľadať v *redukovanom grafe variantov*, ktorý si popíšeme v tejto kapitole. Ukážeme si, že aj keď hľadať cesty v (redukovanom) grafe variantov je náročná úloha (veta 4), nájsť hrany, po ktorých tieto cesty prechádzajú, nie je až také ťažké. Práve tieto hrany budú hranami v redukovanom grafe variantov.

Definícia 5. *Nech* $\{c_1, \ldots, c_k\}$ *je minimálna množina ciest popisujúcich všetky hrany v grafe variantov* G. Redukovaný graf variantov *je zjednotením všetkých hrán použitých cestami* $c_1, \ldots, c_k$.



Obr. 3.1: Redukovaný graf variantov z grafu na obr. 2.2. Pomocou algoritmu popísaného v tejto kapitole sme vybrali hrany, z ktorých sa budú skladať cesty popisujúce všetky hrany v grafe variantov.

Veta 5. *Hrana z grafu variantov sa nachádza v redukovanom grafe variantov práve vtedy, keď leží na niektorej ceste v optimálnom riešení problému 2.*

Dôkaz. Tvrdenie triviálne vyplýva z definície redukovaného grafu variantov. □

Označme $v_1, \ldots, v_n$ vrcholy grafu variantov, v poradí rastúcej pozície v sekvencii. Nasledujúce tvrdenia poukazujú na dôležité vlastnosti grafu variantov, ktoré využijeme pri hľadaní redukovaného grafu variantov.

Definícia 6. *Vrchol u nazývame* dosiahnuteľným *z vrchola v_i, ak existuje cesta z v_i do u. Označme si množinu vrcholov, z ktorých je vrchol v_i dosiahnuteľný ako M_i.*

Veta 6. *V grafe variantov neexistuje vrchol v_i, pre ktorého $k > 1$ predchodcov $v_{i_1}, \ldots, v_{i_k}$ s prislúchajúcimi množinami vrcholov $M_{i_1}, \ldots, M_{i_k}$ platí: $v_{i_1} \in M_{i_2} \wedge v_{i_2} \in M_{i_3} \wedge \cdots \wedge v_{i_k} \in M_{i_1}$.*

Dôkaz. Platnosť tejto vety vyplýva z tvrdenia, že graf variantov je orientovaný acyklický graf. □

Veta 7. *Majme tri vrcholy v_i, v_j, v_k a ich množiny vrcholov M_i, M_j, M_k. Potom platí: ak $v_i \in M_j$ a súčasne $v_j \in M_k$, potom $v_i \in M_k$.*

Dôkaz. Platnosť tejto vety vyplýva z tvrdenia, že dosiahnuteľnosť je tranzitívna vlastnosť. □

Dôsledok 1. *Z viet 6 a 7 vyplýva, že keď množinu vrcholov, z ktorých je vrchol v_i dosiahnuteľný, množinu M_i, zjednocujeme s množinami vrcholov, z ktorých sú dosiahnuteľní jeho predchodcovia, v poradí od predchodcu s najvyšším identifikačným číslom*[3] *po predchodcu s najnižším identifikačným číslom vrchola, potom každá hrana, ktorá rozširuje množinu M_i*[4] *o ďalšie vrcholy, musí ležať na niektorej ceste $c_1, \ldots, c_k$.*

Dôkaz. Podľa vety 7 vieme, že na množinách, z ktorých sú vrcholy dosiahnuteľné, platí tranzitívnosť. Z čoho vyplýva, že ak nájdeme vrchol, ktorý rozširuje množinu vrcholov a z ktorých je daný vrchol dosiahnuteľný, potom musí ležať na niektorej ceste $c_1, \ldots, c_k$. Pretože to znamená, že táto hrana nie je zatiaľ pokrytá žiadnou známou cestou. Súčasne vieme, že tento vrchol je najbližší vrchol, vzhľadom na

[3] Vrcholy sú topologicky usporiadané (napr. v poradí rastúcej pozície).

[4] Ide o hranu, ktorej počiatočný vrchol sa ešte nenachádza v M_i.

vrchol v_i (vrcholy prechádzame v poradí od predchodcu s najvyšším identifikačným číslom po predchodcu s najnižším identifikačným číslom vrchola). Z vety 6 vieme, že tento prístup je korektný. □

Na základe uvedených tvrdení môžeme redukovaný graf variantov zostrojiť nasledovne. Postupne prejdeme graf variantov od najľavejšieho po najpravejší vrchol. Pre každý vrchol v_i si pamätáme množinu M_i, ktorú zostrojíme nasledujúcim spôsobom. Na začiatku obsahuje len svoj vlastný vrchol v_i. Postupne ju zjednocujeme s množinami jeho predchodcov v poradí [5] od predchodcu s najvyšším číslom po predchodcu s najnižším číslom vrchola. Z dôsledku 1 vyplýva, že každá hrana, ktorá rozširuje množinu M_i [6] o ďalšie vrcholy musí ležať na niektorej ceste $c_1, \ldots, c_k$, preto ju zavedieme do redukovaného grafu. Redukovaný graf variantov, pre graf variantov z obrázka 2.2, môžeme vidieť na obrázku 3.1.

Algorithm 1 Algoritmus na hľadanie redukovaného grafu variantov.

```
1: vstup: graf variantov G ▷ vrcholy G sú topologicky usporiadané (napr. v poradí rastúcej pozície)
2: vybrateHrany = {}
3: for i = 1 ... n do
4:     M_i = {v_i}
5:     for j = i − 1 ... 1 ; (v_j, v_i) je hrana v G do        ▷ prechádzame predchodcov vrchola v_i
6:                                                            ▷ v poradí od najvyšších identifikátorov
7:         if v_j ∉ M_i then
8:             M_i = M_i ∪ M_j
9:             vybrateHrany = vybrateHrany ∪{(v_j, v_i)}
10:        end if
11:    end for
12: end for
13: vráť vybrateHrany
```

[5] Vrcholy sú topologicky usporiadané (napr. v poradí rastúcej pozície).

[6] Ide o hranu, ktorej počiatočný vrchol sa ešte nenachádza v M_i.

Veta 8. *Algoritmus na hľadanie redukovaného grafu variantov pre každý vstup vráti po konečnom počte krokov správny výsledok. Označme si počet vrcholov vstupného grafu variantov n. Časová zložitosť algoritmu na hľadanie redukovaného grafu variantov je polynomiálna, konkrétne* $O(n^3)$.

Dôkaz. Prvá časť vety týkajúca sa korektnosti algoritmu priamo vyplýva z dôsledku 1. Druhú časť vety, o časovej zložitosti nášho algoritmu, si ukážeme nasledovne. Pre všetky vrcholy v_i konštruujeme ich množinu vrcholov, z ktorých je vrchol v_i dosiahnuteľný. To robíme tak, že postupne prechádzame všetkých predchodcov vrchola v_i a zjednocujeme M_i s množinami M_p tých predchodcov p, ktorí rozširujú množinu M_i. Pričom zjednotenie má časovú zložitosť $O(n)$. Potom celková časová zložitosť nášho algoritmu je $O(n^3)$. □

V tejto časti sme popísali algoritmus na hľadanie hrán, z ktorých sa budú skladať cesty popisujúce všetky hrany v grafe variantov (pozri obrázok 5.6). Tieto hrany sa využijú na zefektívnenie metódy riešiacej problém 2. Ukázali sme korektnosť tohoto algoritmu. Ďalej sme ukázali, že časová zložitosť tohoto algoritmu je polynomiálna, konkrétne $O(n^3)$. V nasledujúcej kapitole ukážeme, ako využijeme túto redukciu pri riešení problému 2.

Kapitola 4

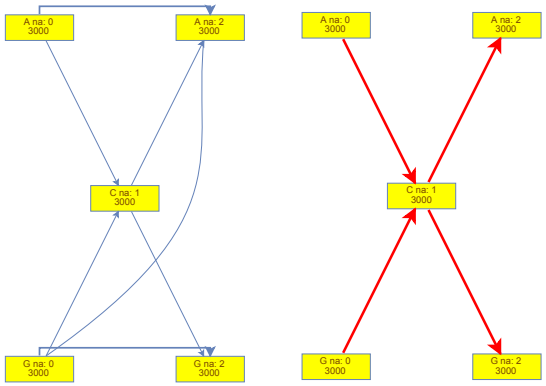
Hľadanie variantov génu

V tejto kapitola čitateľa zoznámime s naším riešením hľadania ciest v (redukovanom) grafe variantov. Keďže ide o NP-ťažký problém (veta 4) navrhujeme ho riešiť pomocou celočíselného lineárneho programovania. Najprv oboznámime čitateľa s neefektívnym riešením, ktoré bude určovať varianty génu priamo z grafu variantov. Následne tento celočíselný lineárny program upravíme tak, aby varianty génov hľadal v redukovanom grafe variantov. S tým, že bude kontrolovať, či nájdené varianty zodpovedajú grafu variantov. Graf variantov bude v tomto riešení hrať úlohu „katalógu prikázaných párov alternatív“ (pozri obrázok 4.1 [7]).

4.1 Hľadanie ciest na grafe variantov

Označme si množinu hrán grafu variantov E a množinu jeho vrcholov V. Ako V_s označme množinu počiatočných vrcholov, vrcholov do ktorých nevchádza žiadna hrana. Podobne V_t označíme množinu koncových vrcholov, teda vrcholov, z ktorých nevychádza žiadna hrana.

[7] Na popísanie všetkých hrán grafu variantov na obrázku sú potrebné tri cesty. Ale ak by hrana medzi Guanínom na nultej pozícii a Adenínom na druhej pozícii nebola, potom by nám stačili aj dva varianty.



Obr. 4.1: Na obrázku vidíme graf variantov (vľavo) a k nemu zodpovedajúci redukovaný graf variantov (vpravo). Obrázok zobrazuje príklad grafu variantov, ktorého hrany ovplyvňujú počet minimálnych ciest popisujúcich všetky jeho hrany.

Predpokladajme, že minimálna množina ciest, ktoré popisujú hrany grafu variantov, má najviac k ciest.

Náš lineárny program obsahuje tri sady binárnych premenných: $x_{i,e}$, $y_{i,e}$ a $z_{i,v}$ s nasledujúcim významom:

1. $x_{i,e} = 1$, ak sa hrana $e \in E$ nachádza na i-tej ceste
2. $y_{i,e} = 1$, ak hrana $e \in E$ je popísaná i-tou cestou
3. $z_{i,v} = 1$, ak vrchol $v \in V$ leží na i-tej ceste

Lineárny program, ktorý bude hľadať cesty na grafe variantov, bude vyzerať nasledovne.

Daný celočíselný lineárny program hľadá k ciest:

$$minimalizuj \sum_{i=1}^{k} \sum_{v \in V_s} z_{i,v}$$

za podmienok [8]:

$$\forall i : \forall (u,v) \in E : \; y_{i,(u,v)} \leq \frac{1}{2}(z_{i,u} + z_{i,v}) \tag{4.1}$$

$$\forall i : \forall v \in V - V_s : \; \sum_{(u,v) \in E} x_{i,(u,v)} = z_{i,v} \tag{4.2}$$

$$\forall i : \forall v \in V - V_t : \; \sum_{(v,w) \in E} x_{i,(v,w)} = z_{i,v} \tag{4.3}$$

$$\forall (u,v) \in E : \; \sum_{i=1}^{k} y_{i,(u,v)} \geq 1 \tag{4.4}$$

$$\forall i : \; \sum_{v \in V_s} z_{i,v} \leq 1 \tag{4.5}$$

$$\forall i : \; \sum_{v \in V_t} z_{i,v} \leq 1 \tag{4.6}$$

$$\forall i : \; \sum_{v \in V} id_v z_{i,v} \geq \sum_{v \in V} id_v z_{i+1,v} \tag{4.7}$$

$$\forall i : \forall (u,v) \in E : \; x_{i,(u,v)} \in \{0,1\} \tag{4.8}$$

$$\forall i : \forall (u,v) \in E : \; y_{i,(u,v)} \in \{0,1\} \tag{4.9}$$

$$\forall i : \forall v \in V : \; z_{i,v} \in \{0,1\} \tag{4.10}$$

- V : množina vrcholov (redukovaného) grafu variantov
- $V_s \subseteq V$: vrcholy, do ktorých nevchádza žiadna hrana
- $V_t \subseteq V$: vrcholy, z ktorých nevychádza žiadna hrana
- E : množina hrán grafu variantov
- $x_{i,e} = 1$: ak hrana $e \in E$ je na i-tej ceste
- $y_{i,e} = 1$: ak hrana $e \in E$ je popísaná i-tou cestou
- $z_{i,v} = 1$: ak vrchol $v \in V$ leží na i-tej ceste
- $x_{i,e}$, $y_{i,e}$, $z_{i,v}$ sú binárne premenné

[8] id_v v siedmej podmienke lineárneho programu je identifikátor vrchola v (topologické poradie rastúcej pozície). Z pohľadu lineárneho programu je tento parameter vrchola konštantou.

Účelovou funkciou chceme zabezpečiť, že si vyberieme minimálny počet ciest. Počet ciest možno spočítať napríklad tak, že spočítame koľkokrát boli použité vrcholy z V_s.

Jednotlivými podmienkami chceme zabezpečiť:

1. podmienkou: aby každá hrana bola priradená k cestám, ktorými je popísaná.
2. podmienkou: jedinečný výskyt vrchola, ktorý nie je štartujúci na každej ceste.
3. podmienkou: jedinečný výskyt vrchola, ktorý nie je končiaci na každej ceste.
4. podmienkou: každá hrana z E sa použije v niektorej ceste.
5. podmienkou: práve jeden štartujúci vrchol pre každú cestu.
6. podmienkou: práve jeden končiaci vrchol pre každú cestu.
7. podmienkou: heuristickú elimináciu riešení s rovnakou hodnotou účelovej funkcie.

Tento lineárny program hľadá minimálny počet ciest v grafe variantov. Súčasne identifikuje varianty génov, čím je na prvý pohľad úloha vyriešená. Ale ak by sme ju použili v praxi, bola by nepoužiteľná.

4.2 Hľadanie ciest na redukovanom grafe variantov

Aby bol náš prístup aplikovateľný na reálnych dáta, navrhli sme hľadať cesty v redukovanom grafe variantov. Graf variantov bude plniť úlohu akýchsi obmedzení, ktoré musia jednotlivé prípustné riešenia spĺňať.

Označme si množinu hrán redukovaného grafu E_r. Nato, aby sme hľadali cesty v redukovanom grafe variantov, musíme modifikovať celočíselný lineárny opísaný v predchádzajúcej podkapitole. Premenné $x_{i,e}$ budeme uvažovať, len pre hrany redukovaného grafu variatov (keďže len tieto hrany sa nachádzajú v optimálnom riešení). Ďalšie zmeny sa odzrkadlia v nasledujúcich modifikáciách podmienok prípustnosti:

- 2. podmienka: $\forall i : \forall v \in V - V_s : \quad \sum_{(u,v)\in E_r} x_{i,(u,v)} = z_{i,v}$
- 3. podmienka: $\forall i : \forall v \in V - V_t : \quad \sum_{(v,w)\in E_r} x_{i,(v,w)} = z_{i,v}$

- 8. podmienka: $\forall i : \forall (u,v) \in E_r : \quad x_{i,(u,v)} \in \{0,1\}$

Tiež treba pridať k hore uvedeným podmienkam takúto podmienku:
$\forall (u,v) \in E_r : \quad \sum_{i=1}^{k} x_{i,(u,v)} \geq 1$

4.3 Výsledný celočíselný lineárny program

Predpokladajme, že minimálna množina ciest, ktoré popisujú hrany grafu variantov, má nanajvýš k ciest. Je veľmi dôležité správne odhadnúť počet ciest. Ak si zvolíme k menšie ako počet ciest, ktoré popíšu všetky hrany grafu variantov, úloha sa stane neprípustnou. To je spôsobené tým, že neexistuje vopred daný počet ciest, ktorý by splnil všetky podmienky celočísleného lineárneho programu. Ak by sme však pracovali s príliš veľkým počtom ciest, potom sa úloha stane z praktického hľadiska výpočtovo veľmi náročnou.

Náš lineárny program obsahuje tri sady binárnych premenných: $x_{i,e}$, $y_{i,e}$ a $z_{i,v}$ s nasledujúcim významom:

1. $x_{i,e} = 1$, ak hrana $e \in E_r$ je na i-tej ceste
2. $y_{i,e} = 1$, ak hrana $e \in E$ je popísaná i-tou cestou
3. $z_{i,v} = 1$, ak vrchol $v \in V$ leží na i-tej ceste

Teda pre každú potenciálnu cestu budeme mať jednu sadu premenných. V prípade, že cesta nie je potrebná, budú všetky premenné, zodpovedajúce danej ceste nulové.

Daný celočíselný lineárny program hľadá k ciest:

$$minimalizuj \sum_{i=1}^{k} \sum_{v \in V_s} z_{i,v}$$

za podmienok [9]:

$$\forall i : \forall (u,v) \in E : \; y_{i,(u,v)} \leq \frac{1}{2}(z_{i,u} + z_{i,v}) \tag{4.11}$$

$$\forall i : \forall v \in V - V_s : \; \sum_{(u,v) \in E_r} x_{i,(u,v)} = z_{i,v} \tag{4.12}$$

$$\forall i : \forall v \in V - V_t : \; \sum_{(v,w) \in E_r} x_{i,(v,w)} = z_{i,v} \tag{4.13}$$

$$\forall (u,v) \in E_r : \; \sum_{i=1}^{k} x_{i,(u,v)} \geq 1 \tag{4.14}$$

$$\forall (u,v) \in E : \; \sum_{i=1}^{k} y_{i,(u,v)} \geq 1 \tag{4.15}$$

$$\forall i : \; \sum_{v \in V_s} z_{i,v} \leq 1 \tag{4.16}$$

$$\forall i : \; \sum_{v \in V_t} z_{i,v} \leq 1 \tag{4.17}$$

$$\forall i : \; \sum_{v \in V} id_v z_{i,v} \geq \sum_{v \in V} id_v z_{i+1,v} \tag{4.18}$$

$$\forall i : \forall (u,v) \in E_r : \; x_{i,(u,v)} \in \{0,1\} \tag{4.19}$$

$$\forall i : \forall (u,v) \in E : \; y_{i,(u,v)} \in \{0,1\} \tag{4.20}$$

$$\forall i : \forall v \in V : \; z_{i,v} \in \{0,1\} \tag{4.21}$$

[9] id_v v ôsmej podmienke lineárneho programu je identifikátor vrchola v (topologické poradie rastúcej pozície). Z pohľadu lineárneho programu je tento parameter vrchola konštantou.

- V : množina vrcholov (redukovaného) grafu variantov
- $V_s \subseteq V$: vrcholy, do ktorých nevchádza žiadna hrana
- $V_t \subseteq V$: vrcholy, z ktorých nevychádza žiadna hrana
- E : množina hrán grafu variantov
- E_r : množina hrán redukovaného grafu
- $x_{i,e} = 1$: ak hrana $e \in E_r$ je na i-tej ceste
- $y_{i,e} = 1$: ak hrana $e \in E$ je popísaná i-tou cestou
- $z_{i,v} = 1$: ak vrchol $v \in V$ leží na i-tej ceste
- $x_{i,e}$, $y_{i,e}$, $z_{i,v}$ sú binárne premenné

Účelovou funkciou chceme zabezpečiť, že vyberieme minimálny počet ciest. Počet ciest možno spočítať napríklad tak, že spočítame koľkokrát boli použité vrcholy z množiny V_s.

Jednotlivými podmienkami chceme zabezpečiť:

1. podmienkou: aby každá hrana bola priradená k cestám, ktorými je popísaná.
2. podmienkou: jedinečný výskyt vrchola, ktorý nie je štartujúci na každej ceste.
3. podmienkou: jedinečný výskyt vrchola, ktorý nie je končiaci na každej ceste.
4. podmienkou: každá hrana z E_r sa použije v niektorej ceste.
5. podmienkou: každá hrana z E je niektorou cestou popísaná.
6. podmienkou: práve jeden štartujúci vrchol pre každú cestu.
7. podmienkou: práve jeden končiaci vrchol pre každú cestu.
8. podmienkou: heuristickú elimináciu riešení s rovnakou hodnotou účelovej funkcie.

Veta 9. *Minimálny počet ciest v grafe variantov, ktoré popisujú všetky jeho hrany práve raz, nie je viacej, než počet všetkých hrán v grafe.*

Dôkaz. Ak tieto cesty spolu popisujú všetky hrany grafu variantov, potom musí každá z nich popisovať aspoň jednu hranu, ktorá nepopisuje žiadna iná cesta, inak by sme ju mohli vynechať. □

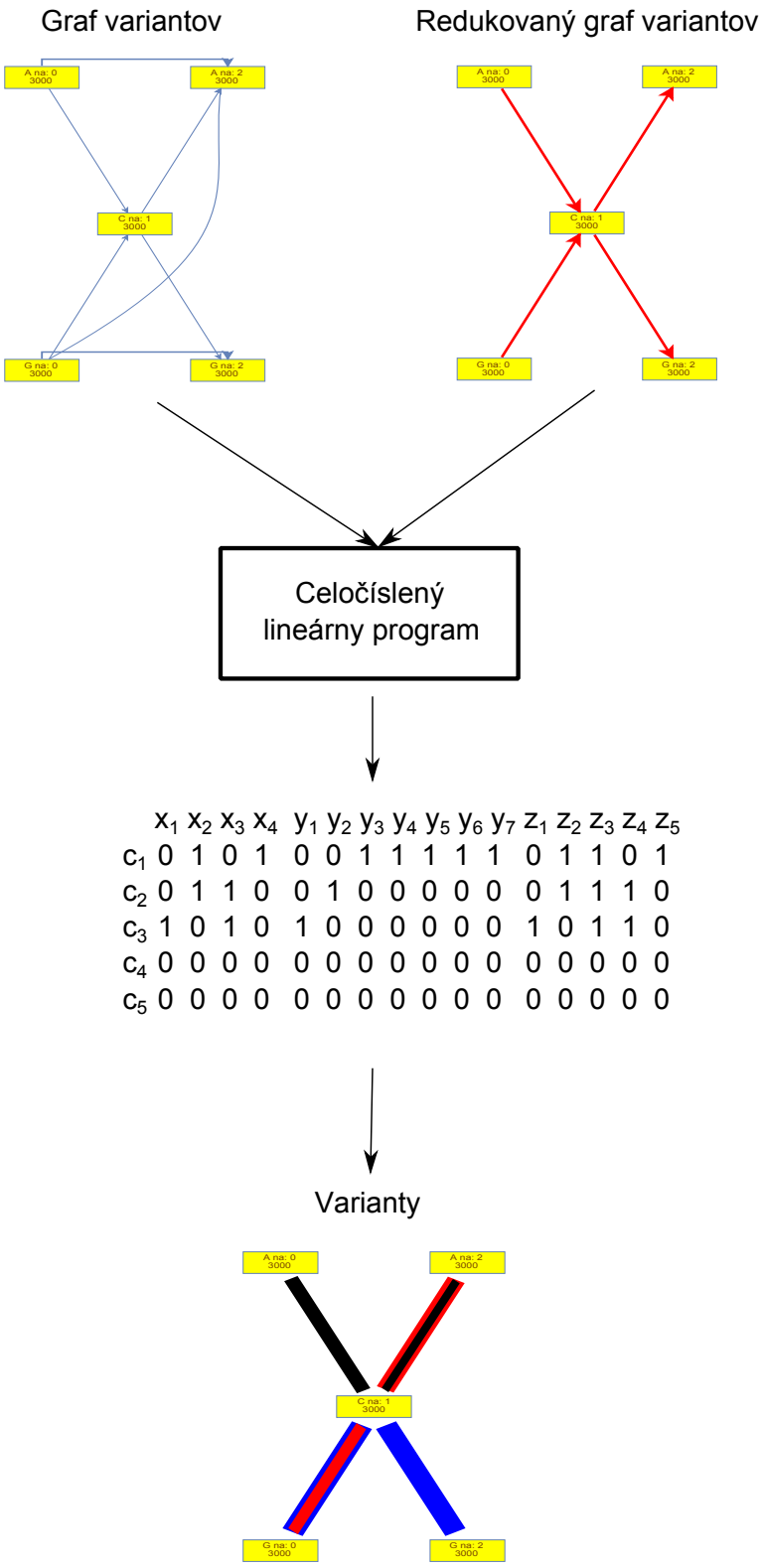
Veta 10. *Označme si počet vrcholov vstupného grafu variant n, počet všetkých hrán vstupného grafu variant m a maximálny počet rôznych ciest v grafe variantov, ktoré popisujú všetky jeho hrany práve raz označme písmenom k. Lineárny program hľadajúci minimálny počet ciest v grafe variantov, ktoré popisujú všetky jeho hrany, má polynomiálne veľa podmienok, konkrétne $O(k + kn + km)$.*

Dôkaz. Vzhľadom na to, že pre každú uvažovanú cestu a pre každú hranu a vrchol na tejto ceste máme najviac osem podmienok, celkový počet podmienok je $O(k + kn + km)$. □

Veta 11. *Označme si počet vrcholov vstupného grafu variantov n a maximálny počet rôznych ciest v grafe variantov, ktoré popisujú všetky jeho hrany práve raz nech je k. Minimalizačná funkcia lineárneho programu hľadajúceho minimálny počet ciest v grafe variantov, ktoré popisujú všetky jeho hrany má polynomiálne veľa premenných, konkrétne $O(kn)$.*

Dôkaz. Keďže pre každý vrchol, z každej uvažovanej cesty, máme jednu premennú, minimalizačná funkcia bude mať $O(kn)$ premenných. □

V tejto kapitole (pozri obrázok 5.6) sme oboznámili čitateľa s vývinom celočíslenného lineárneho programu, ktorý rieši informatické vyjadrenie problému 1 (pozri problém 2). Kapitola sa začne s neefektívnym hľadaním ciest na grafe variantov. Tento prístup sa modifikuje na redukovaný graf variantov, čím sa stane celá metóda aj reálne aplikovateľná (pozri obrázok 4.2). Ukázali sme, že počet ciest, ktoré v ňom uvažujeme, je polynomiálny. Ďalej sme ukázali, že tento lineárny program má polynomiálne veľa podmienok konkrétne $O(k + kn + km)$. Nakoniec sme ukázali, že minimalizačná funkcia tohoto celočíselného lineárneho programu má polynomiálne veľa premenných, konkrétne $O(kn)$.



Obr. 4.2: Ukážka ako funguje nami navrhnutý celočíselný lineárny program. Na základe grafu variantov a redukovaného grafu variantov nájde minimálny počet ciest, ktorými je možné popísať všetky hrany grafu variantov.

Kapitola 5

Výsledky

Náš prístup sme otestovali na dátach, ktoré obsahovali varianty kvasinkového génu formate dehydrogenase. Tento gén je zaujímavý tým, že sa v genóme niektorých kvasiniek nachádza vo viacerých kópiách a tým vytvára viacero variantov génu, napríklad *Saccharomyces cerevisiae* má 2 kópie, *Candida albicans* má 3 kópie, *Yarrowia lipolytica* má 10 kópii.

5.1 Experimentálne dáta

Grafy na obrázkoch 2.2 a 3.1 boli vytvorené z experimentálnych dát časti genómu kvasinky *Magnusiomyces magnusii*, ktorá s veľkou pravdepodobnosťou obsahovala varianty génu formate dehydrogenase. Parametre experimentálnych dát sú uvedené v tabuľke 5.1.

Parametre súboru dát spárovaných segmentov	
Počet spárovaných segmentov	149 388 segmentov
Priemerná dĺžka spárovaných segmentov	100.741 báz ±1.221 bázy
Priemerná veľkosť medzery	60 báz
Parametre referenčného génu	
Dĺžka DNA sekvencie	1150 báz

Tabuľka 5.1: Tabuľka parametrov experimentálnych dát

5.2 Použitý hardware a software

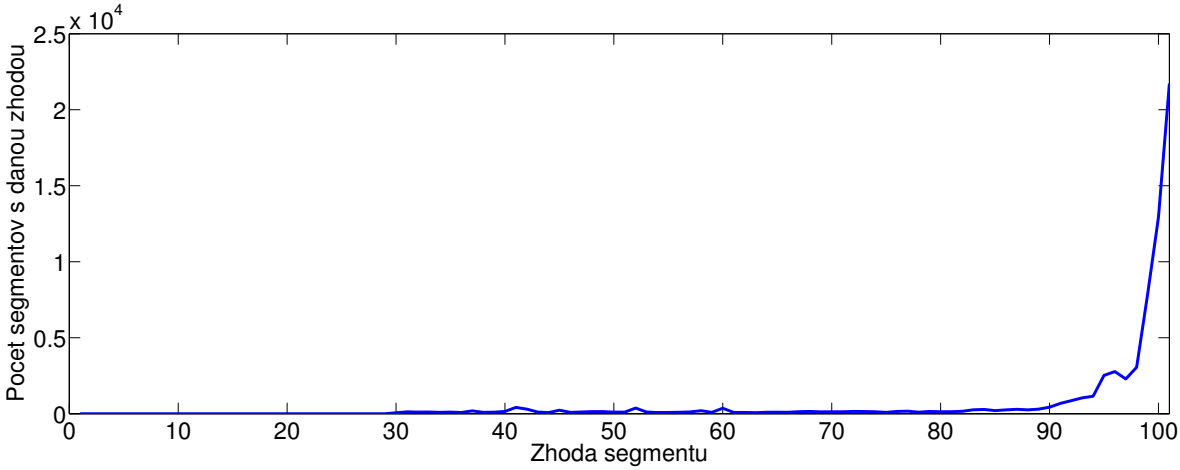
Počítačová zostava, na ktorej prebiehal náš experiment, bol server s ôsmimi jadrami (2 × quad-core Intel Xeon E5520 @ 2.27GHz) s 16 GB RAM. Operačný systém na serveri bolo Ubuntu 12.04.2 LTS. Segmenty sme zarovnali na sekvenciu génu formate dehydrogenase algoritmom Blat [Ken02]. Algoritmy na konštrukciu grafu variantov a redukovaného grafu variantov sme implementovali v jazyku *Java* [10]. Celočíselný lineárny program sme riešili pomocou optimalizačného nástroja *CPLEX*.

5.3 Spracovanie experimentálnych dát

Najprv sme zarovnali vstupné segmenty k sekvencii génu formate dehydrogenase. Od zarovnaných segmentov sme požadovali, aby sa s referenčnou sekvenciou génu zhodovali aspoň s 90 bázami. Tým sme vybrali 74 694 segmentov z pôvodných 149 388 segmentov. Tento parameter sme nastavili na základe obrázku 5.1. Na základe tohto zarovnania sme vytvorili graf variantov tak, ako ho popisujeme v 2. kapitole (pozri obrázok 2.2). Z neho sme odvodili algoritmom 1 opísaným v 3. kapitole redukovaný graf variantov (pozri obrázok 3.1). Následne sme na základe týchto dvoch grafov určili pomocou celočíselného lineárneho programu, popísaného v 4. kapitole jednotlivé varianty génu formate dehydrogenase v kvasinke *Magnusiomyces magnusi*.

[10] Zostrojenie grafu variantov (v tejto fáze prebiehalo aj filtrovanie experimentálnych dát) spolu so zostrojením jeho redukovaného grafu variantov trvalo na tomto hardwari 12 minút.

Maximálny počet uvažovaných ciest sme nastavili na 10.



Obr. 5.1: Graf závislosti počtu rovnakých báz na tých istých pozíciách (zhody) zarovnaného segmentu s referenčnou sekvenciou génu (os x) od počtu vrcholov (os y v logaritmickej škále), ktoré mali danú zhodu.

5.4 Filtrovanie chybných dát

Aby sme mohli naše metódy aplikovať na reálne dáta, museli sme sa zbaviť chybných vrcholov nachádzajúcich sa v grafe variantov. Filtrovanie bolo jedným z najväčších problémov, s ktorým sme sa počas použitia našej metódy na skutočných dátach stretli. Ako sme už spomenuli v 1. kapitole, sekvenovacie technológie produkujú aj chybné segmenty, s čím sme sa museli vysporiadať. Navrhujeme dva typy filtrovania:

- naivné filtrovanie
- percentuálne filtrovanie

Naivné filtrovanie spočíva v stanovení počtu segmentov, ktoré obsahujú daný vrchol v grafe variantov. Určíme si minimálny počet svedkov, ktorí musia dosvedčiť daný vrchol v grafe variantov. Také filtrovanie môže byť v niektorých situáciách nespravodlivé a môže viesť k chybných výsledkom. Napríklad keď nejaký úsek varianty je nedostatočne pokrytý, potom ho takýmto filtrovaním odfiltrujeme celý. Ak by sme však hranicu posunuli nadol, teda zmiernili filtrovanie, potom by sme na dobre pokrytých úsekoch nemuseli odfiltrovať všetky chyby.

Percentuálne filtrovanie spočíva v percentuálnom zastúpení jednotlivých segmentov zarovnaných na danú nejednoznačnú pozíciu, obsahujúcich daný vrchol v grafe variantov. Určujeme minimálne percento svedkov, aké musí mať daný vrchol v grafe variantov.

Nech sa v danom stĺpci, v ktorom chceme uplatniť filtrovanie, nachádza k vrcholov (k je maximálne rovné 4, keďže DNA má 4 rôzne nukleotidy). Každému vrcholu prislúcha určitý počet segmentov, ktoré sú jeho svedkovia. Označme si túto hodnotu pre i-ty vrchol v danom stĺpci ako N_i. Počet všetkých segmentov, ktoré sa zarovnali k danému stĺpcu je $N = \sum_{i=1}^{k}(N_i)$. Potom pre i-ty vrchol v konkrétnom stĺpci filtrujeme podľa pomeru $\frac{N_i}{N}$.

Percentuálne filtrovanie je racionálnejšie než naivné filtrovanie, pretože dokáže lepšie zohľadniť variabilitu pokrytia sekvencie DNA osekvenovanými segmentami.

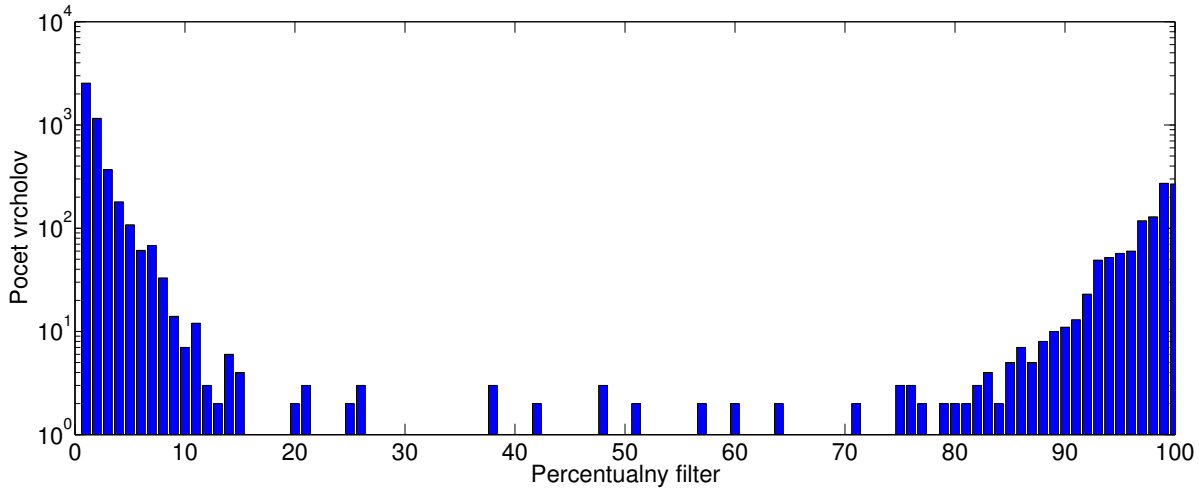
Filtrovanie sme robili na základe počtu segmentov, ktoré obsahovali daný vrchol v grafe variantov. Na jednej strane je dôležité nebrať do úvahy chybné informácie, ale na druhej strane filtrovanie nemôže byť príliš prísne, pretože potom by sme nebrali do úvahy celú informáciu obsiahnutú v segmentoch a výsledky, ktoré by sme takto dostali, by boli neúplné. Z tabuľky 5.2 vidíme, že filtrovanie má výrazný vplyv na výsledný počet variantov, ktoré sme našli na reálnych dátach. V podstate sa dá povedať, že s prísnejším filtrovaním identifikujeme menší počet variantov. To je spôsobné tým, že časom odstránime vrchol, kvôli ktorému sme uvažovali daný variant. Tento jav sa deje až na situáciu, keď sa zväčší počet komponentov grafu variantov. V takejto situácii však nedokážeme povedať, ako vyzerajú celkové varianty. Na základe dát vieme len to, ako vyzerajú varianty v rámci komponentov súvislosti grafu variantov. Keď však chceme zistiť, ako vyzerajú celkové varianty, nezostáva nám nič iné, než konštatovať, že sú kombináciou ciest každého komponentu súvislosti s každým komponentom súvislosti v grafe variantov.

Nasledujúca séria obrázkov (pozri obrázky 5.3, 5.4, 5.5) ukazuje rôzne nastavenia filtrovania vrcholov na tých istých dátach, ako prezentované výsledky (t.j. súbor segmentov variantov génu formate dehydrogenase v kvasinke *Magnusiomyces magnusii*). Na obrázkoch 5.4 a 5.5 vidíme ukážku príliš striktného filtrovania. Takého nastavenie filtrovania spôsobí, že výstupom našej metódy bude nekompletná informácia.

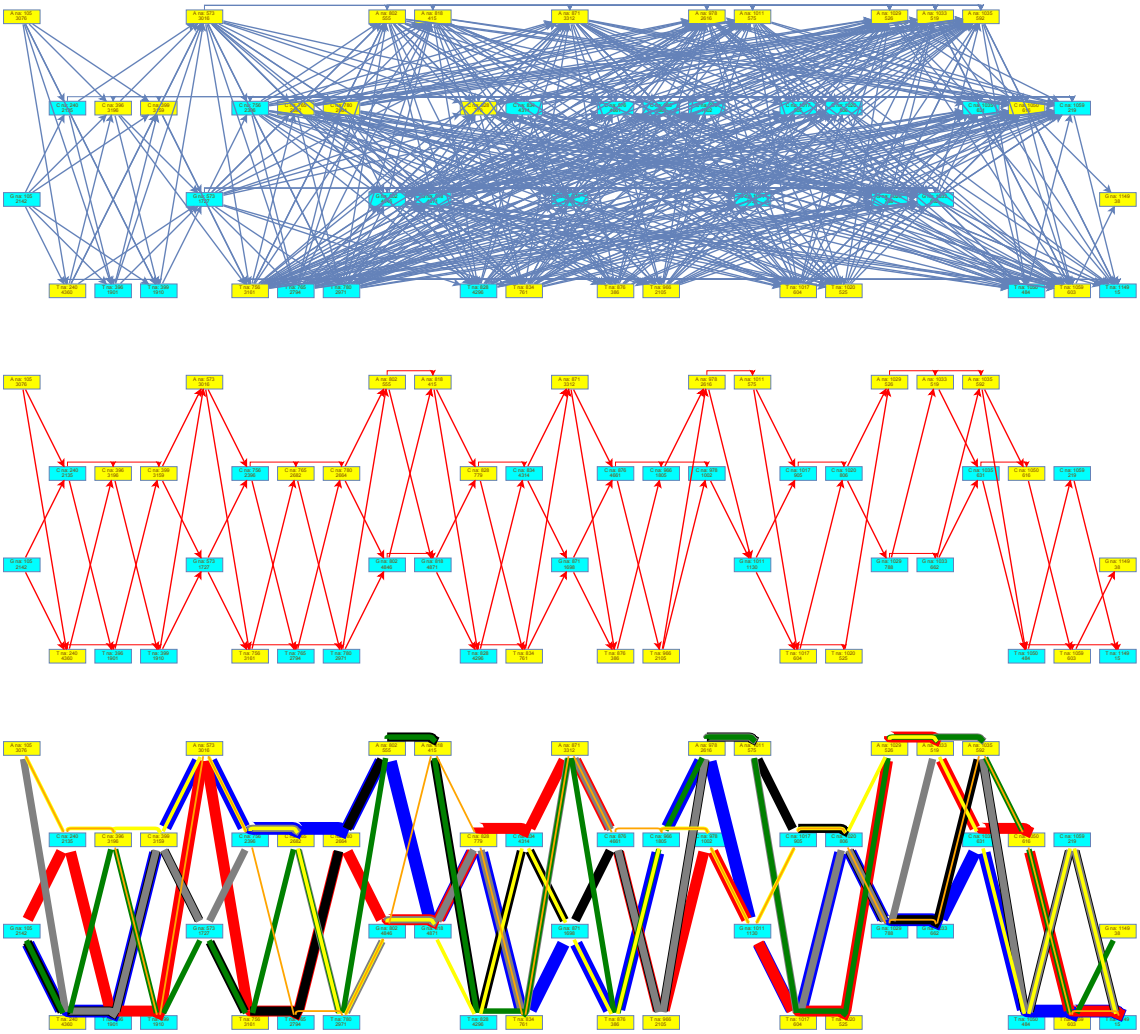
Úroveň filtrovania	Počet identifikovaných variantov	$\|V\|$	$\|E\|$	$\|E_r\|$
25%	7	50	705	89
55%	6	18	131	29
70%	6	10	40	16

Tabuľka 5.2: Vplyv filtrovania na identifikáciu variantov, graf variantov $G = (V, E)$ a jeho redukovanú verziu $G_r = (V, E_r)$.

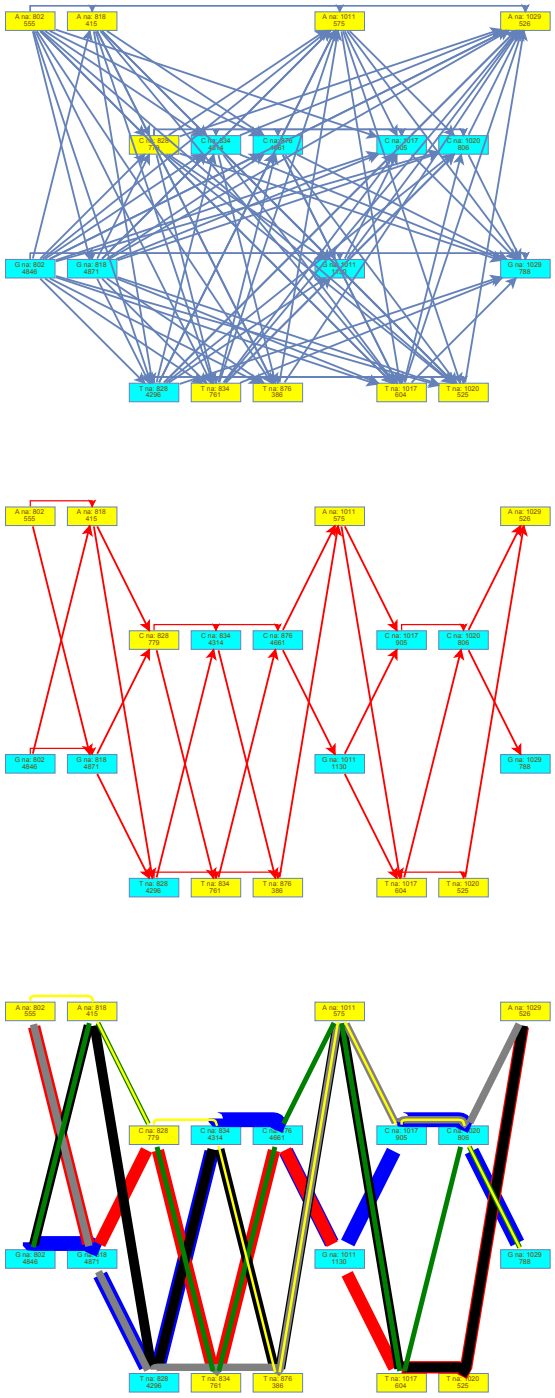
Výsledky dosiahnuté na experimentálnych dátach, pochádzajúcich z genómu kvasinky *Magnusiomyces magnusii*, sú znázornené na obrázku 5.3. Na základe grafu (pozri obrázok 5.2), ktorý vyjadroval závislosť percentuálneho filtrovania od počtu vrcholov, ktoré mali dané percentuálne zastúpenie svedkov, sme sa rozhodli nastaviť filtrovanie vrcholov na 25%.



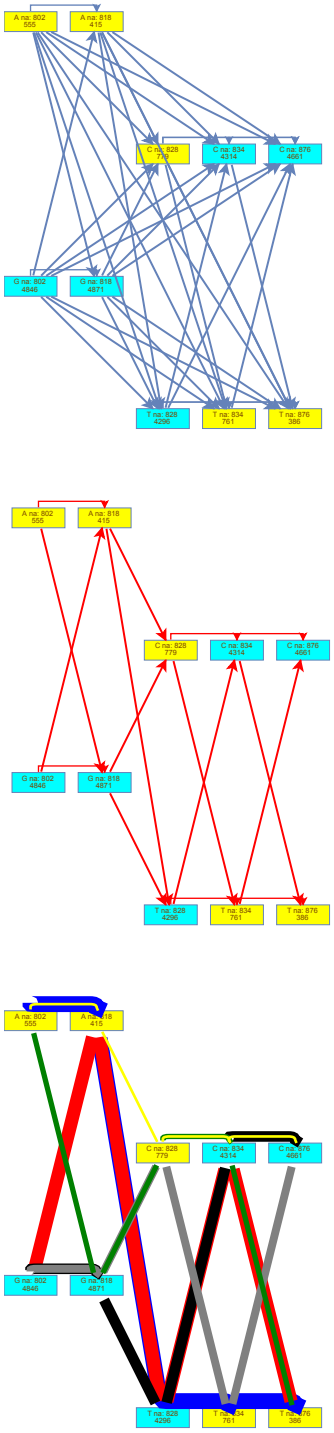
Obr. 5.2: Graf závislosti percentuálneho filtrovania (os x) od počtu vrcholov (os y v logaritmickej škále), ktoré mali dané percentuálne zastúpenie svedkov.



Obr. 5.3: Ako prvý je na nasledujúcich obrázkoch graf variantov, za ním nasleduje redukovaný graf variantov a na konci strany sa nachádzajú výsledné varianty. Percentuálne filtrovanie vrcholov je nastavené na 25 %. Identifikovali sme 7 variantov génu.



Obr. 5.4: Ako prvý je na nasledujúcich obrázkoch graf variantov, za ním nasleduje redukovaný graf variantov a na konci strany sa nachádzajú výsledné varianty. Percentuálne filtrovanie vrcholov je nastavené na 55 %. Identifikovali sme 6 variantov génu.



Obr. 5.5: Ako prvý je na nasledujúcich obrázkoch graf variantov, za ním nasleduje redukovaný graf variantov a na konci strany sa nachádzajú výsledné varianty. Percentuálne filtrovanie vrcholov je nastavené na 70 %. Identifikovali sme 6 variantov génu.

5.4.1 Porovnanie MATLAB-ovskej implementácie s implementáciou v CPLEX-e

Ukazuje sa, že na vyriešenie tohoto problému je potrebné mať k dispozícii dobre zoptimalizovaný ILP solver. Dôkazom toho sú naše problémy, ktoré nás sprevádzali počas prvej implementácie. Vtedy sme sa pokúšali použiť ILP solver v *Matlab*-e. Táto implementácia však neviedla k úspešnému nasadeniu našej metódy (pozri tabuľku 5.3). *Matlab* nielenže bol schopný v rozumnom čase vypočítať optimálne riešenie (do 24 hodín), ale dokonca jeho ILP solver nesprávne orezával vetvy v strome prípustných riešení. V dôsledku čoho nevrátil rovnaké výsledky ako *CPLEX*.

		Matlab		**CPLEX**	
		Čas optimalizácie	Výstup	Čas optimalizácie	Výstup
Úroveň filtrovania	25%	977 sekúnd	neprípustnosť	1534.84 sekúnd	7 variantov
	55%	13 sekúnd	neprípustnosť	175.87 sekúnd	6 variantov
	70%	∞	–	8.54 sekúnd	6 variantov

Tabuľka 5.3: Porovnanie implementácie celočíselného lineárneho program v *Matlab*-e s implementáciou v *CPLEX*-e.

Nevýhodou *CPLEX*-u bolo zdĺhavé predspracovanie (vytvorenie vstupného súboru, ktorý reprezentuje konkrétny celočíselný lineárny program), ktoré *Matlab* nevyžadoval. Pri úrovni filtrovania nastavenej na 25% trvalo 277 minút, na 55% trvalo 65 minút a na 70% trvalo 11 minút.

5.5 Simulované experimenty

Vierohodnosť našej metódy sme sa rozhodli overiť pomocou experimentov, ktoré sme implementovali v jazyku *Java*. Cieľom prvého experimentu bolo zistiť ako vplýva veľkosť referenčnej sekvencie na identifikáciu variantov. Cieľom druhého experimentu bolo preskúmať vplyv počtu substitúcií na identifikáciu variantov. Simulátor má 6 parametrov:

1. dĺžka referenčnej sekvencie (zároveň ide aj o dĺžku variantov)
2. počet variantov

3. počet substitúcií v jednom variante
4. konzistentnosť variantov s referenčnou sekvenciou - pravdepodobnosť tranzície
5. pravdepodobnosť vyradenia hrany grafu variantov
6. maximálna vzdialenosť susedných vrcholov grafu variantov

V simuláciách predpokladáme, že výsledný graf variantov neobsahuje chybné vrcholy.

5.6 Priebeh simulácií

Na začiatku simulácie sa náhodne zvolí sekvencia referenčného génu s danou dĺžkou. Potom sa začne generovať daný počet variantov nasledovným spôsobom. Náhodne sa určí miesto [11] v referenčnom géne, na ktorom prebehne substitúcia. Substitúcia prebehne tak, aby sa zachovala konzistentnosť výsledného variantu voči sekvencii referenčného génu. Takýmto spôsobom sa odsimuluje daný počet substitúcií. Keď máme vygerované varianty, vytvárame graf variantov. Najprv vytvoríme vrcholy grafu variantov, potom pridáme medzi tieto vrcholy hrany. Hranu pridáme s určitou pravdepodobnosťou [12] medzi tie vrcholy, ktoré sú v požadovanej vzdialenosti. Z takto získaného grafu variantov sme odvodili algoritmom 1 opísaným v 3. kapitole redukovaný graf variantov. Potom sme na základe týchto dvoch grafov určili pomocou celočíselného lineárneho programu jednotlivé varianty simulovaného génu.

[11] Toto je možno až veľmi prísny predpoklad. Ako vidíme na grafe variantov, ktorý vznikol na reálnych dátach, tieto miesta neboli až tak náhodné. Vidíme, že graf obsahuje 25 substitúcií, ale pritom celkový počet vrcholov je len 50. V našich simuláciách zodpovedá tejto situácii simulácia, v ktorej je dĺžka referenčnej sekvencii 30 báz. Na dĺžku referenčnej sekvencie sa totiž dá pozerať aj ako na počet potenciálnych miest, na ktorých prebehne s veľkou pravdepodobnosťou mutácia. Keďže presne nepoznáme ako a na akých miestach vznikajú génové mutácie, tak sme sa ich rozhodli simulovať takýmto modelom. Tento náš prístup spôsobil náročnejšiu identifikáciu variantov.

[12] Hrany medzi vrcholmi idúcimi za sebou, vo variante, sme pridali vždy. V opačnom prípade by nebola splnená veta 3.

5.7 Vplyv veľkosti referenčnej sekvencie na identifikáciu jej variantov

V tejto simulácii sme overovali ako ovplyvní dĺžka referenčnej sekvencie a teda aj dĺžka jednotlivých variantov počet variantov, ktorý je schopná identifikovať naša metóda. V tomto experimente sme zafixovali počet substitúcií v jednom variante na hodnotu 15. Skúmali sme varianty s dĺžkou 30, 350, 700 a 1050 báz. Cieľom v týchto experimentoch bolo identifikovať 3, 4, 5, 6 a 7 variantov. Dosiahnuté výsledky sú prezentované v tabuľke 5.4.

Dĺžka sekvencie variantov	Počet variantov	$\|V\|$	$\|E\|$	$\|E_r\|$	Počet identifikovaných variantov
30 báz	3	44	413	56	3
30 báz	4	52	668	72	4
30 báz	5	53	626	79	5
30 báz	6	57	815	86	–
30 báz	7	50	656	76	6
350 báz	3	58	419	77	3
350 báz	4	76	655	98	4
350 báz	5	98	481	133	5
350 báz	6	115	480	153	6
350 báz	7	125	589	180	5

Dĺžka sekvencie variantov	Počet variantov	$\|V\|$	$\|E\|$	$\|E_r\|$	Počet identifikovaných variantov
700 báz	3	85	614	114	3
700 báz	4	115	630	159	4
700 báz	5	139	688	191	4
700 báz	6	164	793	230	–
700 báz	7	188	795	269	5
1050 báz	3	86	472	111	3
1050 báz	4	119	541	168	4
1050 báz	5	144	609	201	5
1050 báz	6	168	514	235	4
1050 báz	7	204	575	292	4

Tabuľka 5.4: Tabuľka dosiahnutých výsledkov z prvého experimentu (použitý znak „–“ v tabuľke znamená, že ILP solver neskončil v časovom limite 24 hodín).

Z tabuľky 5.4 sa môžeme presvedčiť, že dĺžka sekvencie trochu ovplyvňuje počet variantov, ktorý je naša metóda schopná identifikovať. Simulácie ukazujú, že problém, ktorý treba vyriešiť, je časová náročnosť. Totiž v situácii, keď nastala chyba, sa ako jej dôsledok javí nedostatočný počet hrán, ktorý by dokázal rozlíšil daný počet variantov. To však znamená pracovať so zložitejším grafom variantov a riešiť náročnejší celočíselný lineárny program. Preto sme sa rozhodli identifikovať daný počet variantov na jednoduchších grafoch variantov. Toto naše rozhodnutie spôsobilo, ako vidieť v tabuľke 5.4, identifikáciu menšieho počtu variantov.

5.8 Vplyv počtu substitúcií na identifikáciu variantov génu

V tejto simulácii sme overovali ako vplýva počet substitúcií v jednotlivých variantoch na počet variantov, ktorý je naša metóda schopná identifikovať. Z predchádzajúceho experimentu totiž vyplynulo, že pri dlhých sekvenciách je nevyhnutné, aby sme pracovali so zložitými grafmi. V tomto experimente nás zaujímalo, či tento jav dokáže nejakým spôsobom ovplyvniť počet substitúcií. Preto sme si v tomto experimente zafixovali dĺžku referenčnej sekvencie a teda aj dĺžku variantov na 1050 báz. Skúmali sme varianty s počtom substitúcií 5, 10, 15, 20 a 25. Cieľom v týchto experimentoch bolo identifikovať 4, 5, 6 a 7 variantov. Dosiahnuté výsledky sú prezentované v tabuľke 5.5.

Počet substitúcií v variante	Počet variantov	$\|V\|$	$\|E\|$	$\|E_r\|$	Počet identifikovaných variantov
5	4	40	305	52	4
5	5	49	203	66	5
5	6	60	353	86	6
5	7	70	449	97	5
10	4	79	762	109	4
10	5	100	647	143	5
10	6	118	664	164	–
10	7	139	684	199	–

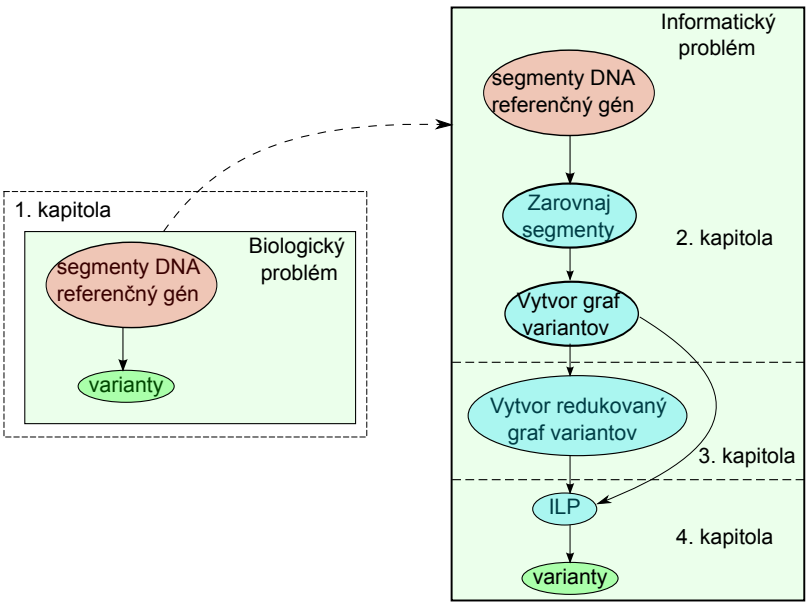
Počet substitúcií v variante	Počet variantov	$\|V\|$	$\|E\|$	$\|E_r\|$	Počet identifikovaných variantov
15	4	119	541	168	4
15	5	144	609	201	5
15	6	168	514	235	4
15	7	204	575	292	4
20	4	155	659	206	4
20	5	187	625	256	4
20	6	225	614	317	4
20	7	273	1045	387	4
25	4	194	797	271	4
25	5	232	657	326	5
25	6	278	989	387	5
25	7	327	810	463	4

Tabuľka 5.5: Tabuľka dosiahnutých výsledkov z druhého experimentu (použitý znak „–“ v tabuľke znamená, že ILP solver neskončil v časovom limite 24 hodín).

Náš experiment ukázal, že počet substitúcií len do istej miery ovplyvní identifikáciu variantov. Pri malom počte substitúcií sa dá očakávať, že varianty budú natoľko jedinečné, že dokážeme ich identifikovať na základe našej metódy. Keď počet substitúcií pribúda, narastá aj zložitosť grafu variantov. V podstate sa dá povedať, že graf variantov do istej miery reprezentuje pokrytie sekvencie segmentami. Je však zrejmé, že s nízkym pokrytím nedokážeme zostaviť kompletný genóm. Takisto s jednoduchým grafom variantov nedokážeme identifikovať všetky varianty.

Záver

V diplomovej práci sme skúmali, ako sa dajú pomocou spárovaných segmentov zostaviť varianty toho istého génu. Najprv sme si tento problém charakterizovali ako biologický problém (pozri problém 1). Tento problém sme následne pretransformovali na informatický problém (pozri problém 2). Na to, aby sme mohli vyriešiť informatický problém na zostavovanie variantov toho istého génu sme si vytvorili grafovú reprezentáciu jednotlivých alternatív voči referenčnému génu. Túto štruktúru nazývame graf variantov. Každý variant génu je v tomto grafe vyjadrený cestou z počiatočného vrchola do koncového vrchola. V dôsledku aplikovania tejto metódy sme museli zaviesť istý predpoklad na dáta, ktorý väčšina reálnych dát spĺňa. Na tomto grafe sme sa pokúsili identifikovať varianty génov, ale riešenie sa ukázalo pre praktické účely ako nevyhovujúce. Preto sme sa rozhodli graf variantov zredukovať kubickým algoritmom na redukovaný graf variantov. Pomocou celočísleného lineárneho programu sme hľadali minimálny počet ciest vedúcich po redukovanom grafe variantov, ktorý popisuje každú hranu grafu variantov (pozri obrázok 5.6). Na záver sme tento vylepšený prístup otestovali na experimentálnych dátach pochádzajúcich z kvasinky *Magnusiomyces magnusii* a na simulovaných dátach.



Obr. 5.6: Obrázok schematicky zachytáva myšlienkový proces identifikácie variantov génov z dát sekvenovania novej generácie popísaný v tejto práci.

Náš prístup ku novému problému však má určité obmedzenia.

Na reálnych dátach môže nastať situácia, keď varianty skúmaného génu majú určité alternatívy na začiatku, ako aj na konci sekvencie. V takomto prípade nám osekvenované segmenty neposkytujú dostatočnú informáciu, v dôsledku čoho sa graf variantov a aj jeho redukovaná verzia rozpadnú na komponenty súvislosti. V tomto prípade použitie našej metódy identifikuje varianty na začiatku a konci, avšak je veľmi ťažké určiť, ako budú vyzerať celkové sekvencie variantov. Takouto situáciou sme sa v práci nezaoberali. Pracovali sme iba so súvislými grafmi.

V našej práci uvažujeme spárované segmenty, ktoré nie sú veľmi vzdialené. Medzera medzi nimi je menšia ako dĺžka segmentov. V literatúre sa takéto segmenty označujú pojmom *pair-end reads*. Druhý typ spárovaných segmentov má páry s veľmi veľkou vzdialenosťou. Ide o tzv. *mate – pair reads*. Zostáva otvorenou otázkou ako by sa dali v našej metóde využiť takéto segmenty.

V prvej kapitole sme uviedli, že existujú tri základné typy génových mutácií: substitúcie, inzercie a delécie. V práci sa však venujeme len substitúciám. V budúcnosti by bolo možné podobné metódy vytvoriť aj pre ďalšie typy génových mutácií.

V praxi sa môže stať, že lineárny program si bude musieť vybrať medzi viacerými optimálnymi riešeniami. Bolo by zaujímavé zistiť, či by počet segmentov zarovnaných

k daným nejednoznačným pozíciám nedokázal určiť lepšie riešenie, pretože je zrejmé, že počet zarovnaných segmentov závisí od počtu variantov daného génu.

Určiť korektne počet variantov môže tiež pomôcť sekvencia z okolia referenčnej sekvencie génu. Touto informáciou by sa dali teoreticky rozlišovať jednotlivé varianty génu.

Literatúra

[BNA+12] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A. Gurevich, Mikhail Dvorkin, Alexander S. Kulikov, Valery M. Lesin, Sergey I. Nikolenko, Son Pham, Andrey D. Prjibelski, Alexey V. Pyshkin, Alexander V. Sirotkin, Nikolay Vyahhi, Glenn Tesler, Max A. Alekseyev, and Pavel A. Pevzner. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology : a journal of computational molecular cell biology*, 19(5):455–477, May 2012.

[CBP09] Mark J. Chaisson, Dumitru Brinza, and Pavel A. Pevzner. De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome Research*, 19(2):336–346, February 2009.

[CJB10] Miklós Csürös, Szilveszter Juhos, and Attila Bérces. Fast mapping and precise alignment of ab solid color reads to reference dna. In *Proceedings of the 10th international conference on Algorithms in bioinformatics*, WABI'10, pages 176–188, Berlin, Heidelberg, 2010. Springer-Verlag.

[DEKM99] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, July 1999.

[FM01] Paolo Ferragina and Giovanni Manzini. An experimental study of an opportunistic index. In *In Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 269–278, 2001.

[GMS80] John Gallant, David Maier, and James A. Storer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20(1):50–58, February 1980.

[Hal06] Yancey Hall. Coming Soon: Your Personal DNA Map? *National Geographic News*, 2006.

[KBBV13] Martin Kravec, Martin Bobák, Broňa Brejová, and Tomáš Vinař. Variants of Genes from the Next Generation Sequencing Data. 2013.

[KDB+06] Lutz Krause, Naryttza N. Diaz, Daniela Bartels, Robert A. Edwards, Alfred PŁhler, Forest Rohwer, Folker Meyer, and Jens Stoye. Finding novel genes in bacterial communities isolated from the environment. *Bioinformatics*, 22(14):e281–e289, 2006.

[Ken02] W James Kent. Blat—the blast-like alignment tool. *Genome research*, 12(4):656–664, 2002.

[LLK08] Ruiqiang Li, Yingrui Li, Karsten Kristiansen, and Jun Wang 0004. Soap: short oligonucleotide alignment program. *Bioinformatics*, 24(5):713–714, 2008.

[LRD08] Heng Li, Jue Ruan, and Richard Durbin. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858, 2008.

[LTPS09] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3):R25–10, March 2009.

[LZZ+08] Hao Lin, Zefeng Zhang, Michael Q. Zhang, Bin Ma, and Ming Li. Zoom! zillions of oligos mapped. *Bioinformatics*, 24(21):2431–2437, 2008.

[MDH11] Andre Minoche, Juliane Dohm, and Heinz Himmelbauer. Evaluation of genomic high-throughput sequencing data generated on illumina hiseq and genome analyzer systems. *Genome Biology*, 12(11):R112, 2011.

[MPC+11] Paul Medvedev, Son Pham, Mark Chaisson, Glenn Tesler, and Pavel Pevzner. Paired de bruijn graphs: a novel approach for incorporating mate pair informationinto genome assemblers. *Journal of computational biology : a journal of computational molecular cell biology*, 18(11):1625–1634, 2011.

[NVP12] Francesca Nadalin, Francesco Vezzi, and Alberto Policriti. Gapfiller: a de novo assembly approach to fill the gap within paired reads. *BMC Bioinformatics*, 13(Suppl 14):S8, 2012.

[PAS+13] Son K. Pham, Dmitry Antipov, Alexander Sirotkin, Glenn Tesler, Pavel A. Pevzner, and Max A. Alekseyev. Pathset graphs: a novel approach for comprehensive utilization of paired reads in genome assembly. *Journal of computational biology : a journal of computational molecular cell biology*, 20(4):359–371, April 2013.

[PC12] Pierre Peterlongo and Rayan Chikhi. Mapsembler, targeted and micro assembly of large NGS datasets on a desktop computer. *BMC bioinformatics*, 13(1):48, 2012.

[PS08] Mihai Pop and Steven L. Salzberg. Bioinformatics challenges of new sequencing technology. *Trends in Genetics*, 24(3):142 – 149, 2008.

[PSL+11] Nicolas Philippe, Mikael Salson, Thierry Lecroq, Martine Leonard, Therese Commes, and Eric Rivals. Querying large read collections in main memory: a versatile data structure. *BMC Bioinformatics*, 12(1):242, 2011.

[PTW01] P A Pevzner, H Tang, and M S Waterman. An eulerian path approach to dna fragment assembly. *Proc Natl Acad Sci USA*, 98(17):9748–9753, 2001.

[SF77] Coulson AR. Sanger F, Nicklen S. DNA sequencing with chain terminating inhibitors. *Proc. Natl Acad. Sci. USA*, 74(12):5463 – 5467, 1977.

[TPS09] Cole Trapnell, Lior Pachter, and Steven L. Salzberg. Tophat: discovering splice junctions with rna-seq. *Bioinformatics*, 25(9):1105–1111, 2009.

[WGF10] John C. Wooley, Adam Godzik, and Iddo Friedberg. A primer on metagenomics. *PLoS Comput Biol*, 6(2):e1000667, 02 2010.

[WH11] Rene L. Warren and Robert A. Holt. Targeted assembly of short sequence reads. *PLoS One*, 6(5):e19816, 2011.

[ZB08] Daniel R. Zerbino and Ewan Birney. Velvet: Algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 18(5):821–829, 2008.

[ZRS+12] Zhiyuan Zhai, Gesine Reinert, Kai Song, Michael S. Waterman, Yihui Luan, and Fengzhu Sun. Normal and Compound Poisson Approximations for Pattern Occurrences in NGS Reads. *Journal of Computational Biology*, 19(6):839–854, June 2012.

Príloha A

CD so zdrojovými súbormi

Na CD sa nachádzajú nasledovné súbory:

- zdrojové súbory programu generujúceho graf variantov a redukovaný graf variantov
- zdrojové súbory celočíselného lineárneho programu (verzia v CPLEX-e aj v Matlab-e)
- zdrojové súbory simulátora
- elektronická verzia diplomovej práce vo formáte pdf