Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5220-52026

Bc. Tomáš Farkaš

Zhlukovanie a analýza veľkých dátových súborov

Diplomová práca

Vedúci práce: doc. RNDr. Mária Lucká, PhD.

Máj 2017

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

FIIT-5220-52026

Bc. Tomáš Farkaš

Zhlukovanie a analýza veľkých dátových súborov

Diplomová práca

Študijný program: Softvérové inžinierstvo
Študijný odbor: 9.2.5 Softvérové inžinierstvo
Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU, Bratislava
Vedúci práce: doc. RNDr. Mária Lucká, PhD.

Máj 2017

ČESTNÉ PREHLÁSENIE

Čestne prehlasujem, že záverečnú prácu som vypracoval samostatne, s použitím uvedenej literatúry a na základe svojich vedomostí a znalostí.

.....
Bc. Tomáš Farkaš

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

ZHLUKOVANIE A ANALÝZA VEĽKÝCH DÁTOVÝCH SÚBOROV

Študijný program: Softvérové inžinierstvo
Autor: Bc. Tomáš Farkaš
Vedúci práce: doc. RNDr. Mária Lucká, PhD.
Máj 2017

Práca skúma zhlukovanie veľkých dátových súborov v oblasti bioinformatiky, pri ktorom sa využijú metódy spracovania signálov na zistenie podobnosti reťazcov DNA.
Teoretická časť práce rozoberá typy zhlukovacích algoritmov, teóriu matematických transformácií z domény spracovania signálov, ktoré sa majú použiť na spracovanie sekvencií DNA, skúma tiež špecifiká týchto sekvencií. Ďalej sa zaoberá prácami iných autorov v podobnej oblasti a tiež modifikáciami generických porovnávacích funkcií do oblasti bioinformatiky.
V ďalšej časti navrhuje nové algoritmy, ktoré zavádzajú princípy a postupy spracovania dát na základe znalostí o ich štruktúre. Jedná sa o použitie matematických transformácií, ktoré doteraz na sekvencie DNA použité neboli a tiež o prístup založený na analýze s viacerými rozlíšeniami.
Časť venujúca sa implementácii skúma existujúce softvérové prostriedky z oblasti dátovej analytiky a ich nedostatky. Navrhuje tiež zlepšenia ich architektúry a realizácie. Ďalej popisuje implementáciu komponentov, ktoré vykonávajú zmieňovanú analýzu DNA sekvencií.
Posledná časť porovnáva presnosť navrhnutých metód tak s etablovanými, ako aj experimentálnymi metódami z dostupnej literatúry, s cieľom potvrdiť alebo vyvrátiť teoretické vlastnosti metód vyplývajúce z ich charakteristík.

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

DEEP ANALYSIS AND CLUSTERING OF BIG DATA FILES

Degree Course: Software Engineering
Author: Bc. Tomáš Farkaš
Supervisor: doc. RNDr. Mária Lucká, PhD.
May 2017

The thesis explores the clustering of large data files in the field of bioinformatics, which utilizes signal processing methods for comparing sequences of DNA data.
The first section discusses the types of clustering algorithms, the theory of mathematical transformations to be applied for the comparison functions improvement and the sequence characteristics in general. Then it deals with the work of other authors in the use of mathematical transformations for string comparison as well as the modification of generic comparison functions for the field of bioinformatics.
The next part proposes new methods which process the data using basic knowledge of their internal structure. This include utilising signal processing transformations never before used for DNA data and multiple resolution analysis methods.
The third part describes implementation of the algorithms. First, examining the implementation of existing software tools in the field of data analytics and their drawbacks. It also proposes improving their architecture and realization. Further describes the implementation of components that perform the mentioned analysis of DNA sequences.
The last section compares the accuracy of the proposed methods with both established and experimental methods from the literature available to confirm or refute the theoretical behavior of the methods resulting from their properties.

PREDSLOV

Za cenné rady, pripomienky a usmerňovanie pri vypracovávaní mojej diplomovej práce ďakujem mentorke doc. RNDr. Márii Luckej, PhD.

OBSAH

ZOZNAM OBRÁZKOV

Časť I

TEORETICKÉ ZÁKLADY PRÁCE

1

ÚVOD

Ostatné roky sú v oblasti biológie a bioinformatiky charakteristické rozmachom sekvenačných technológií novej generácie, tzv. NGS *(Next Generation Sequencing)*. Tieto umožnili aj menším pracoviskám, za dostupné náklady, tvoriť veľké množstvá dát, ktoré ale je potrebné ďalej spracovávať počítačovými systémami.
Typickou úlohou spracovania takýchto dát je korekcia chýb a skladanie krátkych načítaných sekvencií, tzv. *short reads* do dlhších *contigov*. Ich vytvorením sa síce získajú reťazce DNA, no tie, ako také, nič nehovoria o vlastnostiach tkaniva, z ktorého boli získané. Ďalším krokom je preto zisťovanie vlastností tkanív cez hľadanie vzorov v dátach. Práca sa zameriava na hlbšiu analýzu sekvencií s cieľom porovnať a zhlukovať ich podľa vzájomnej podobnosti. Za týmto účelom chceme, na rozdiel od súčasných metód zakladajúcich sa na štatistických vlastnostiach dát, použiť prostriedky z domény spracovania signálov.

2

ÚVOD DO TEÓRIE ZHLUKOVANIA DÁT

Zhlukovaním dát (klastrovaním) sa rozumie ich zoskupovanie do zhlukov tak, aby medzi entitami v rovnakom zhluku navzájom bola väčšia podobnosť, ako voči entitám z iného zhluku [9]. Táto všeobecná formulácia pokrýva veľké množstvo algoritmov, z ktorých každý má špecifické vlastnosti a je ho možné použiť na určitú množinu dát.

2.1 ZÁKLADNÉ DELENIE ALGORTIMOV ZHLUKOVANIA DÁT

Keďže predošlá definícia pokrýva veľkú množinu problémov, má význam proces zhlukovania dát došpecifikovať pre rôzne typy úloh, vstupov a/alebo požadovaných výstupov.

2.1.1 *Distribučné zhlukovanie dát*

Distribučné zhlukovanie dát [9, 29, 6] *(angl. Partitioning clustering methods, distributive clustering)* vytvára jednoduché jednoznačné zhluky. Platí, že každý zhluk obsahuje minimálne jeden prvok a každý prvok patrí práve do jedného zhluku. Formálne definované: Majme množinu dát s n prvkami: $D = \{e_1, ..., e_n\}$. Distribučné zhlukovanie rozdelí tieto prvky do k zhlukov $C_1, ..., C_k$, pričom platí: $\forall i \leqslant k : C_i \in D$ a súčasne $\forall i, j \leqslant k, i \neq j : C_i \cap C_j = \emptyset$.
Známe algoritmy *(k-means, k-medoids)* začnú s k náhodne vytvorenými zhlukmi, ktoré postupne iteratívne optimalizujú sledujúc zlepšenie (zníženie hodnoty) funkcie rozdielnosti prvkov v rámci zhluku. Jasnou nevýhodou algoritmov spĺňajúcich vyššie dané podmienky je (1.) Jednoznačné priradenie prvkov do zhlukov a (2.) Závislosť od vopred stanoveného parametra k. Okrem toho negarantujú nájdenie globálneho optima.

2.1.2 *Hierarchické zhlukovanie dát*

Algoritmy hierarchického klastrovania [9, 29, 6] vytvárajú hierarchiu zhlukov, tzv. dendrogram. Tento typ klastrovania sa aplikuje na dáta, u ktorých je možné hierarchiu identifikovať, t.j. majú viacúrovňové kritériá podobnosti, napríklad v bioinformatike fylogenetický strom dedenia organizmov. Sú známe dva prístupy: Aglomeratívny prístup (zdola nahor) spája drobné zhluky reláciou podobnosti do nadskupín, ktoré neskôr pokryjú celú množinu objektov. Menej častý distributívny prístup (zhora nadol) delí väčšie zhluky do menších, v ktorých sú si prvky viac podobné. Z povahy algoritmov vyplýva ťažkopádna zmena vetvenia stromovej štruktúry. Pokiaľ algoritmus v jednom kroku chybne zlúči [rozdelí] dve vetvy, na nápravu je potrebné znovu generovať celé delenie. Situáciu komplikuje aj zlá schopnosť detekcie chybného kroku. Algoritmus je schopný po čase zistiť, že kvalita zhlukov je nízka, ale nie je schopný určiť, v ktorom kroku chyba nastala.

Aglomeratívne prístupy typicky udržujú v každom kroku vzdialenostnú maticu vrcholov, z ktorej sa po spojení dvoch vrcholov - vetiev tieto odstránia a matica sa prepočíta obsahujúc nový vrchol reprezentujúci aglomerát dvoch vetiev pôvodných. Existujú viaceré prístupy, ktoré sa líšia počítaním vzdialenosti ľubovoľného vrcholu x od vrcholu m vytvoreného spojením dvoch vetiev s_1 a s_2:

- Jednoduché spojenie *(angl. Single Linkage - SLINK)*[24]:
  $d(x, m) = min(d(x, s_1), d(x, s_2))$
- Úplné spojenie *(angl. Complete Linkage - CLINK)*[24]:
  $d(x, m) = max(d(x, s_1), d(x, s_2))$
- Párová metóda s aritmetickým priemerom - neváhovaná *(angl. Unweighted Pair Group method with Mean Average - UPGMA)*[8]:
  $d(x, m) = \frac{d(x,s_1),d(x,s_2)}{2}$

2.1.3 *Zhlukovanie dát založené na hustote*

Predošlé metódy vykonávajú nad celou množinou dát operácie ako optimalizácia stredov zhlukov, zmena príslušnosti prvkov k zhlukom, spájanie a delenie zhlukov. Zhlukovanie založené na hustote [9, 29, 6] pracuje sledujúc nasledovné kroky:

1. Nájdi v priestore (n-rozmernom Euklidovskom) oblasti s vyššou frekvenciou výskytu entít. Tieto oblasti prehlás za centrá budúcich zhlukov a prvky zaraď do novovytvorených zhlukov.
2. Prehľadávaj ostatné prvky hľadajúc také, ktoré sú blízke k už zaradeným prvkom. Pokiaľ, spolu s už zaradenými prvkami, ich počet na jednotku priestoru nedosiahne podprahovú hodnotu (a teda hustota je stále vyššia ako vopred definovaný prah), zaraď prvok do zhluku.
3. Pokiaľ sú všetky prvky buď zaradené do zhlukov alebo ich vzdialenosť od akéhokoľvek zhluku zabezpečuje podprahovú hodnotu hustoty, skonči. Prvky sú zaradené do zhlukov, nie nevyhnutne unikátne, nezaradené prvky prehlás za chybné alebo zašumené.

Je zrejmé, že algoritmy sú schopné nachádzať zhluky nekonvexných tvarov. Časová náročnosť prehľadávania priestoru sa dá vylepšiť zaindexovaním prvkov do n-rozmernej mriežky, čím sa vytvorí nová skupinu algoritmov - Zhlukovanie na mriežke *(angl. Grid-based clustering.)*

2.2 FUNKCIE PODOBNOSTI PRVKOV

V časti 2.1.1 bola spomenutá funkcia rozdielnosti prvkov v rámci zhluku *(angl. within-cluster variation)*. Zhlukovanie dát je problém z kategórie tzv. *učenia bez učiteľa*, kedy nie sú k dispozícií trénovacie dáta. Jediným kľúčom na určenie podobnosti prvkov v zhluku, resp. príslušnosti prvku k zhluku je potom vhodne vytvorená metrika podobnosti prvkov. Jej vhodná voľba má teda značný vplyv na kvalitu zhlukovania a má význam sa ňou zaoberať. Konkrétny popis funkcií použiteľných v oblasti bioinformatiky sa nachádza v časti 4.3, táto kapitola sa zaoberá používanými funkciami všeobecne.

2.2.1 *Metriky na určenie kvality zhluku*

Zhlukovanie sa snaží zaradiť prvky do skupín tak, aby medzi prvkami v skupine bola čím väčšia a medzi prvkami rôznych skupín čím menšia

podobnosť [9]. Z tohto vyplývajú dve základné metriky: podobnosť, $sim()$ a rozdielnosť, $dist()$. So vzájomným vzťahom[1]:

$$sim(e_1, e_2) = 1 - dist(e_1, e_2)$$

kde e_1, e_2 sú dva prvky, pre ktoré sa rozdielnosť [podobnosť] vyhodnocuje.
Ďalej je možné vyhodnotiť priemernú rozdielnosť prvkov vrámci zhluku alebo častejšie priemernú rozdielnosť prvkov k centru zhluku:

$$E_{C_1} = \frac{\sum_{p \in C_i} dist(p, c_i)}{n}$$

kde C_i je zhluk, pre ktorý sa metrika určuje, c_i je jeho centrum a n počet prvkov, ktoré obsahuje.
Súčet takýchto zhlukových charakteristík je možné použiť na ohodnotenie celkovej kvality zhlukovania.

2.2.2 *Funkcie rozdielnosti prvkov*

Funkcia rozdielnosti prvkov, označovaná tiež skrátene ako vzdialenostná funkcia, je funkcia o dvoch vstupných parametroch - prvkoch, ktorá kvantifikuje rozdielnosť medzi nimi. Požadujú sa nasledovné vlastnosti [28]:

- **Pozitivita**: $dist(e_1, e_2) \geqslant 0; dist(e_1, e_2) = 0 \iff e_1 = e_2$.
- **Symetria**: $dist(e_1, e_2) = dist(e_1, e_2)$
- **Trojuholníková nerovnosť**: $dist(e_1, e_2) + dist(e_2, e_3) \geqslant dist(e_1, e_3)$.

Prvky sú charakterizované vektorom atribútov. Pre numerické atribúty $a_1 ... a_k$ je funkcia dist typicky Euklidovská alebo Manhattanská vzdialenosť:

$$dist_E(e_1, e_2) = \sqrt[2]{|a_1(e_1) - a_1(e_2)|^2 + ... + |a_k(e_1) - a_k(e_2)|^2}$$

$$dist_M(e_1, e_2) = |a_1(e_1) - a_1(e_2)| + ... + |a_k(e_1) - a_k(e_2)| = \sqrt[1]{|a_1(e_1) - a_1(e_2)|^1 + ... + |a_k(e_1) - a_k(e_2)|^1}$$

Zovšeobecnením je tzv. Minkowského vzdialenosť [33] s parametrom q:

$$dist_q(e_1, e_2) = \sqrt[q]{|a_1(e_1) - a_1(e_2)|^q + ... + |a_k(e_1) - a_k(e_2)|^q}$$

1 Pri normovaní rozsahov všetkých atribútov na rozsah $< 0, 1 >$.

Pre $q = 1$ vzťah počíta Mannhattanskú vzdialenoť, pre $q = 2$ Euklidovskú vzdialenoť, pre $q = \infty$ riešenie limitne konverguje k maximu rozdielu medzi atribútmi, čo je možné, samozrejme, vypočítať aj efektívnejším spôsobom.
Pre nenumerické atribúty je najčastejším postupom ich prevod na numerické atribúty. Týmto sa umožní použitie štandardných vzdialenostných funkcií. Je možné ale tiež navrhnúť špecializovanú vzdialenostnú funkciu pre iné typy dát. V práci sa budeme ďalej zaoberať obidvomi prístupmi.

3

MATEMATICKÉ MODELY A TRANSFORMÁCIE POUŽITÉ V PRÁCI

3.1 FOURIEROVA TRANSFORMÁCIA

Fourierov rad [10] predstavuje aproximáciu ľubovoľných (zložitých) periodických funkcií súčtom jednoduchých periodických funkcií $sin(x)$ a $cos(x)$ - s rozličnou periódou a amplitúdou. Funkciu $f(x)$ je teda možné podľa tejto aproximácie vyjadriť ako:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \Big(a_n sin(nx) + b_n cos(nx) \Big)$$

pričom a_n a b_n sú koeficientami pre $f(x)$

Použitím *Eulerovej identity* $e^{ix} = cos(x) + isin(x)$ je možné ukázať, že:

$$cos(x) = \frac{e^{ix} + e^{-ix}}{2}$$

a

$$sin(x) = \frac{e^{ix} - e^{-ix}}{2i}$$

Potom

$$cos(nx) = \frac{e^{inx} + e^{-inx}}{2}$$

a

$$sin(nx) = \frac{e^{inx} - e^{-inx}}{2i}$$

Keďže (v množine reálnych čísel) $e^{ix} = -e^{-ix}$, je možné súčet kosínusoíd pre ľubovoľnú funkciu vyjadriť ako súčet zložiek e^{inx}:

$$f(x) = \sum c_n e^{inx}$$

Fourierova transformácia (ďalej DFT) je zovšeobecnenie rovnakého princípu pre neperiodické funkcie. Ľubovoľnú funkciu považuje za súčet periodických kmitov rôznych násobkov základnej frekvencie - s rozličnou periódou, amplitúdou a fázovým posunom začiatku. Z vektora

vzoriek (funkčných hodnôt $f(x)$) dĺžky N získa pre každú frekvenčnú zložku k amplitúdu a fázový posun začiatku, pričom sa uvažuje N frekvencií v rozsahu $< 0Hz; N-1Hz > X_k$:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi\frac{n}{N}} \tag{1}$$

kde:
N je dĺžka vektora vzoriek x; x_n je hodnota vektora x v bode $n, x_n \equiv x[n]$, $k \in < 0, N-1 >$ je index v spektrálnej doméne
X_k je hodnota amplitúdy a fázového posunu začiatku pre zložku o frekvencií k. Vo výsledku zložka pre fázu počiatku je:

$$phs_k = tan^{-1}\left(re(X_k), im(X_k)\right)$$

čo predstavuje uhol osi x a spojnice komplexného výsledku a počiatku v dvojrozmernej koordinátovej sústave komplexných čísel. Zložka pre amplitúdu je:

$$amp_k = \sqrt{re(X_k)^2.im(X_k)^2}$$

čo predstavuje absolútnu veľkosť komplexného výsledku.
Algoritmus DFT teda transformuje vektor reálnych čísel dĺžky n na vektor komplexných čísel dĺžky n uchovávajúci informácie o zložkách frekvenčného spektra pre transformovaný signál.
Fourierova transformácia sa využíva najmä v spracovaní signálov, pri ktorých je potrebné rozlíšiť rôzne frekvencie alebo zistiť rozloženie frekvenčného spektra. Aplikácie zahŕňajú typicky separáciu rádiových frekvencií pri príjme, odstránenie šumu, alebo oddelenie spektier zvukových signálov za účelom zosilnenia alebo zoslabenia ich zložiek.

3.1.1 *Alternatívne maticové vyjadrenie DFT*

Koreň jednotky (de Moivreovo číslo) je komplexné číslo, ktoré umocnené prirodzeným číslom N sa rovná 1.

$$\omega_n = (cos\frac{2\pi n}{N} + isin\frac{2\pi n}{N}) = e^{-i2\pi\frac{n}{N}}$$

Zložky vo vzťahu pre výpočet DFT sú teda korene jednotky. Súčin matice M_n takej, že $(M_n)_{j,k} = \omega^{(i-1)(j-1)}$ a vstupného vektora x je rovný výstupnému vektoru X, ktorý je výsledkom DFT.

$$M_4 = \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 \\ \omega^0 & \omega^2 & \omega^4 & \omega^5 \\ \omega^0 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix}$$

Obr. 1: Vzorová matica DFT pre $N = 4$; $\omega_4 = i$

3.1.2 *Rýchla Fourierova transformácia*

Algoritmus DFT, podľa (1) počíta pre každú zložku frekvenčného spektra X_k; $k =< 0, N-1 >$ súčet N hodnôt. Časová náročnosť algoritmu teda dosahuje $O(n^2)$. Na dlhé vstupné vektory (viď 4.3) sa tým stáva nepoužiteľným.
Efektívnejší spôsob výpočtu, s náročnosťou $O\big(nlog(n)\big)$ je známy ako Cooleyho-Tukeyho algoritmus [4], resp. Rýchla Fourierova Transformácia (FFT).
Algoritmus je rekurzívny, typu *divide and conquer* (rozdeľuj a panuj):

1. FFT jednoprvkového vektora sa rovná jemu samému.
2. Vstupný vektor x dĺžky N rozdeľ na dva vektory $x^{(1)}, x^{(2)}$, polovičnej dĺžky.
3. Rekurentne zisti $X^{(1)}, X^{(2)}$ ako FFT polovičných vstupných vektorov $x^{(1)}, x^{(2)}$.
4. Výsledok komponuj pomocou koreňov jednotky takto:
   pre $i =< 0, \frac{N}{2})$:
$$X_i = X_i^{(1)} + \omega_N^i X_i^{(2)}$$
$$X_{i+\frac{N}{2}} = X_i^{(1)} - \omega_N^i X_i^{(2)}$$

Algoritmus je z dôvodu delení na polovičné vektory efektívny pre dĺžky 2^n (n je prirodzené číslo). Vektor inej dĺžky sa doplní nulovými hodnotami na najbližšiu väčšiu mocninu.

3.2.1 *Walshova-Hadamardova transformácia*

Walshova-Hadamardova transformácia (WHT) [21] je zovšeobecnením Fourierovej transformácie. Prvky obdobnej transformačnej matice sa počítajú vzťahom:

$$(H_n)_{j,k} = \frac{1}{n}(-1)^{j.k}$$

j.k je skalárny súčin bitových reprezentácií čísel j, k.

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Obr. 2: Vzorová Walshova matica pre $N = 4$

Aj pre WHT existuje rekurzívny algoritmus rýchlej transformácie (FWHT) [21]. Ako Obr. 2 uvádzame vzorovú maticu pre transformáciu 4-prvkového vektora. V literatúre sa označuje buď ako H_4 alebo ako H_2, keďže transformácia sa z rovnakých dôvodov používa na vektory dĺžky 2^n, v tomto prípade $n = 2$.
Sú dostupné práce zaoberajúce sa použitím WHT [20] v oblasti počítačového videnia na vyhľadávanie a identifikáciu (dvojrozmerných) obrazových vzorov, čo je úloha podobná porovnávaniu jednorozmerných reťazcov.

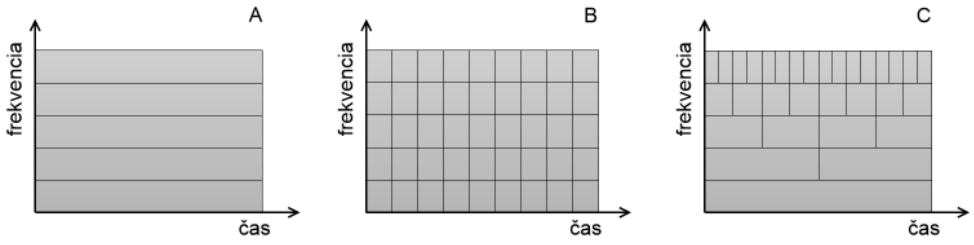
3.2.2 *Vlnkové transformácie (Haarova transformácia)*

Vlnkové transformácie *(angl. Wavelet transformation, WT)* sú transformácie, ktoré zo vstupného signálu generujú časovo a frekvenčne závislé výstupy. Pracujú na princípe okna, do ktorého je vpísaná wavelet funkcia. Okno (veľkosti menšej, nanajvýš rovnej ako vstupný vektor) prechádza cez hodnoty vektora a postupne integruje súčiny wavelet funkcie a hodnôt vstupného vektora. Pre malé hodnoty frekvencie sa používa široké okno, ktorým sa zisťuje prítomnosť frekvenčnej zložky naprieč

celým vstupom. Pre rýchlejšie frekvencie sa veľkosť okna zužuje, čím je možné zistiť silu frekvenčnej zložky v rámci vymedzenej časti vstupu.

Rozdiely oproti Fourierovej transformácii:

- Pri Fourierovej transformácii vplývajú hodnoty vstupu na všetky hodnoty výstupu (Fourierove funkcie sú lokalizované pre frekvencie, nie v čase, resp. priestore). Vlnkové transformácie sú lokalizované aj pre frekvencie aj pre čas.
- Niektoré funkcie, napríklad tie s ostrými hranami (čo je aj prípad transformácií znakových reťazcov - viď 4.3), je možné rovnako kvalitne aproximovať využitím menšieho počtu základných wavelet funkcií ako základných sínusoíd, keďže tieto prirodzene opisujú zaoblené dráhy.



Obr. 3: Porovnanie výstupov FT, lokalizovanej FT a WT.

Obrázok 3 porovnáva výstupy Fourierovej (Obrázok 3 - A), lokalizovanej Fourierovej (Obrázok 3 - B) a vlnkovej (Obrázok 3 - C) transformácie. Fourierova transformácia zisťuje prítomnosť frekvencie (a fázy) v rámci celého vstupu. Výsledkom je teda vektor hodnôt síl frekvenčných zložiek. Lokalizovaná Fourierova transformácia pracuje rovnako, no v tomto prípade sa výpočet rozdelí na viacero behov a každý beh dostane ako vstup iba časť vstupného vektora. Týmto sa získa matica hodnôt sily frekvenčnej zložky naprieč frekvenciami a pozíciou v rámci vstupu (časom). Vlnková transformácia z dôvodu postupnej zmeny veľkosti okna poskytne obraz o spektre ako bol popísaný vyššie.

Haarova transformácia

Haarova transformácia je druh vlnkovej transformácie, pričom sa ako základná wavelet funkcia používa tzv. Haarova funkcia definovaná nasledovne:

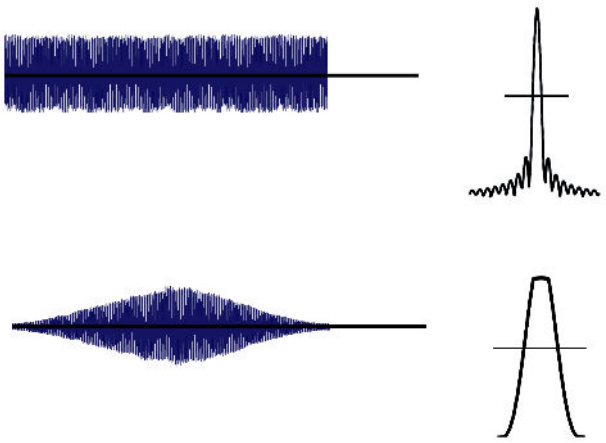
$$\phi(t) = \begin{cases} 1 & \text{pre } 0 \leqslant t < 1/2 \\ -1 & \text{pre } 1/2 \leqslant t < 1 \\ 0 & \text{inak} \end{cases}$$

3.3 ZAHLADZOVANIE V SPEKTRÁLNEJ DOMÉNE

Pri rýchlom spôsobe výpočtu transformácií (FFT, FWHT) sa predpokladá vstupný vektor dát o dĺžke 2^n, kde n je prirodzené číslo. Pokiaľ dáta takúto dĺžku nemajú, je obvyklá praktika doplniť vstupný vektor na takúto dĺžku nulovými hodnotami, ktoré by nemali vplývať na výsledok. Obdobná situácia nastáva aj pokiaľ dáta majú dĺžku $l = 2^n$, ale chceme získať vyššie rozlíšenie spektra, teda viac spektrálnych koeficientov ako l, kedy sa dáta doplnia nulovými hodnotami na dĺžku rovnú želanému počtu spektrálnych koeficientov.
Problém tohto prístupu však spočíva v skokovej zmene vo vektore vstupných dát. Fourierova transformácia aproximuje hodnotu funkcie reprezentovanej vektorom vstupných dát sumou funkcií sínusového tvaru. Bude teda transformovať aj túto skokovú zmenu, napríklad sériou:

$$\sin(x) + \frac{1}{3}\sin(3x) + \frac{1}{5}\sin(5x) + ...$$

Táto séria sa premietne do výstupu a zapríčiní efekt označovaný ako *Gibbsov jav* [13]



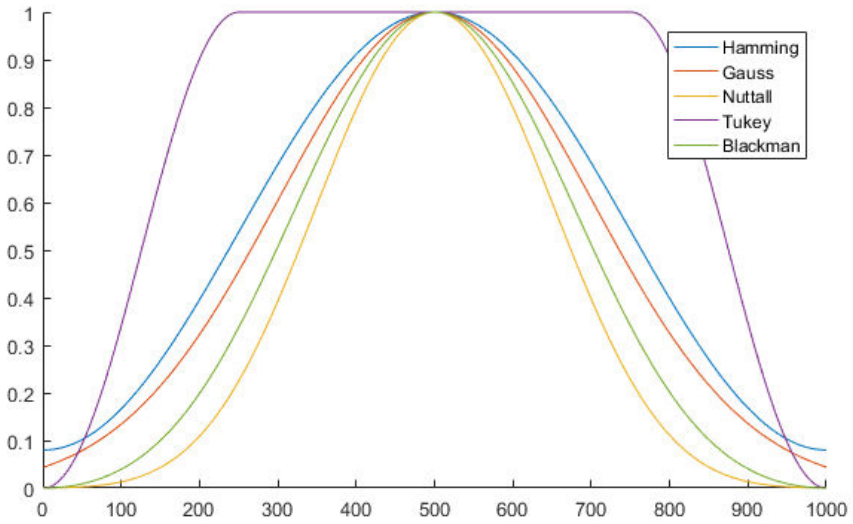
Obr. 4: Porovnanie výstupu transformácie periodického signálu bez (hore) a s použitím zahladzujúceho okna (dole).

Na elimináciu javu sa relevantná (nenulová) časť vstupného vektora prenásobí vektorom hodnôt, ktorý zahladí hrany. Je známych a v literatúre[13, 11] popísaných veľa typov takýchto okien, viď Obr.5. Ich vlastnosti a vplyv sa dajú zhrnúť nasledovne:

- Prvky počiatku a konca vstupného vektora sa násobia menšími hodnotami ako prvky v strede vektora. Ich vplyv na výsledné spektrum je teda menší. Toto je nežiadúci jav, ktorému nedokážeme zabrániť.
- Prvky vstupného vektora sa násobia typicky hodnotami menšími, nanajvýš rovnými jednej. Suma hodnôt prvkov vstupného vektora sa teda zníži a absolútne hodnoty zložiek frekvenčného spektra prestanú byť relevantné. Pri vyhodnocovaní spektier vzniknutých transformáciu bioinformatických dát by jav nemal mať vplyv na výsledok, keďže všetky výstupy budeme normalizovať. Je však potrebné dbať na to, aby sa normalizácia vykonávala podľa charakteristík spektra, napríklad podľa súčtu jeho zložiek a nie podľa charakteristík vstupného vektora.
- Použitie okna zahladí šum vo frekvenčnom spektre, no čiastočne zarovná aj dominantnú zložku spektra - viď Obr.4, na ktorom má dominantná zložka spektra po transformácii s použitím zahladzujúceho okna viditeľne menej ostrý priebeh a zrezaný vrchol *(angl. scalloping)*. Dáta, na ktoré chceme transformácie použiť sú známe častými *inserciami* a *deléciami*, ktoré ovplyvňujú vzdia-

lenosti medzi časťami sekvencií. Takto môžu vzniknúť mierne odchýlky v spektrálnej reprezentácii, kedy sa rozdiely vo vzdialenosti podobných častí genómu reprezentujú mierne odlišnými spektrálnymi koeficientami. Pri počítaní transformácií, využijúc zahladzovacie okná, budeme schopní nájsť prekryv medzi dominantnými spektrálnymi koeficientami (neskôr) aj napriek očakávanému miernemu vzájomnému posunutiu ich pozícií.

Na zvýraznenie poslednej menovanej pozitívnej vlastnosti požadujeme, aby zahladzovacie okno dosahovalo maximálny šírkový rozptyl lokálnych maxím, čomu zodpovedá Blackmannovo-Nutallovo okno (BNW) [7].



Obr. 5: Porovnanie tvaru zahladzujúcich okien veľkosti 1000 podľa viacerých autorov.

4

ZHLUKOVANIE DÁT V BIOINFORMATIKE

Bioinformatika je oblasť informatiky zaoberajúca sa komplexným spracovávaním dát získaných z molekúl DNA. Kapitola poskytne stručný popis používaných pojmov a vykonávaných procesov v oblasti informačného spracovania DNA.

4.1 ZÁKLADNÉ POJMY V BIOINFORMATIKE

4.1.1 *DNA*

Deoxyribonukleová kyselina (DNA), je prírodný polymér prítomný v živých organizmoch. Skladá sa z dvoch závitov špirály zloženej z nukleotidov, medzi ktorými sú väzby z dusíkatých báz - Adenínu, Cytozínu, Guanínu a Tymínu. Informácia nesená molekulou DNA je kódovaná poradím báz a v bioinformatike sa reprezentuje ako znakový alebo bitový reťazec. Bázy sú teda kódované buď znakmi {A, C, G, T} alebo bitovo, kedy sa každému znaku priradí 2-bitový kód.

4.1.2 *Read*

Read je krátka sekvencia, ktorá vznikla procesom sekvencovania (digitalizácie reťazcov) DNA. Jej dĺžka závisí od použitej technológie, typicky sa pohybuje od 100 do 1000 znakov (*base pairs - bp*).

4.1.3 *Contig*

Contig je sekvencia znakov (báz DNA), ktorá vznikla porocesom spájania *read-ov* do dlhších postupností. Jej dĺžka je niekoľko tisíc bp. Contig je vstupom do ďalšieho spracovania reťazcov algoritmami zhlukova-

nia dát. Termínom *sekvencia DNA*, resp. *vstupný vektor*, budeme ďalej označovať práve *contig*.

4.2 POROVNÁVANIE REŤAZCOV DNA

4.2.1 *Metódy založené na zarovnaní reťazcov*

V bioinformatike sa *zarovnávaním reťazcov* DNA [RNA, proteínov] *(angl. Alignment-based methods for sequence analysis)* myslí nájdenie spoločného začiatku dvoch reťazcov a ich preorganizovanie vsunutím alebo odstránením častí reťazcov tak, aby sa rovnaké časti reťazcov mapovali na seba. Cieľom je identifikácia regiónov s rovnakými postupnosťami báz [aminokyselín], ktoré implikujú podobné vlastnosti. Iným využitím je mapovanie načítaných reťazcov na referenčnú sekvenciu. Je pritom potrebné zohľadniť určité javy, ktoré sa v sekvenciách vyskytujú buď prirodzene alebo vznikli chybou pri procese sekvenovania:

- *Insercia*: Vsunutie reťazca dĺžky n do časti genómu. Vo výsledku sa javí ako časť reťazca, ktorú treba pri zarovnaní odstrániť, resp. na jej miesto vsunúť prázdne znaky do referenčnej sekvencie.
- *Repetícia*: $n-$ násobné načítanie časti genómu do výsledného reťazca. Vo výsledku sa javí ako viac replík sekvencie, no nie nevyhnutne za sebou. Pri zarovnaní je potrebné redundantnú repliku odstrániť alebo vsunúť prázdne znaky do referenčnej sekvencie.
- *Delécia*: Odstránenie časti genómu, vo výslednom reťazci podsekvencia chýba. Pri zarovnaní je potrebné doplniť prázdne znaky do sekvencie tak, aby časť pred a po delécií zarovnala s referenčnou sekvenciou.
- *Substitúcia*: Nahradenie, resp. chybné identifikovanie bázy [aminokyseliny] v čítanej sekvencii. Znaky treba nahradiť inými.

Problematika chybovosti je omnoho zložitejšia [1] a nie je predmetom tejto práce, cieľom bolo poukázať na množinu problémov, ktoré musia algoritmy zarovnávania riešiť.
Párové alebo viacnásobné zarovnávanie reťazcov bolo základom metód analýzy sekvencií [19]. Tieto algoritmy sú ale pomerne pomalé, keďže určenie zarovnania resp. vzdialenosti medzi dvomi sekvenciami sa vykonáva párovo a pritom sa berú do úvahy všetky znaky obidvoch sek-

vencií. Sú známe algoritmy (napr. Rabin-Karpov [16]) ktoré vykonajú hľadanie podreťazcov s časovou náročnosťou $O(n + k)$, kde n je dĺžka dátového reťazca a k dĺžka vzorky, $n >> k$. Operácia sa ale vykonáva pre všetky podreťazce, teda $O(n)$ vzoriek a celkový čas spravovania narastie na druhú mocninu dĺžky sekvencií [19]. Je teda zrejmé, že algoritmy so zarovnávaním sekvencií nie sú optimálne pre prácu s dlhými reťazcami a je vhodné sa im vyhnúť, pokiaľ je to možné.

4.2.2 *Metódy nevyžadujúce zarovnanie reťazcov*

Typickým bioinformatickým problémom je proces tvorby tzv. *fylogenetických stromov*. Tým sa myslí hierarchická štruktúra, reprezentujúca vzťahy medzi biologickými druhmi. Pri jeho tvorbe sa pracuje so sekvenciami DNA, ktoré prejdú procesom hierarchického zhlukovania. Na vyhodnotenie podobnosti dvoch biologických druhov nie je však potrebné poznať presné zarovnanie sekvencií, postačuje schopnosť kvantifikovať vzájomnú podobnosť dvoch dát.
Problém kvadratickej časovej náročnosti zarovnávania sekvencií rieši množstvo algoritmov porovnávania reťazcov bez ich zarovnania *(angl. Alignment-free Sequence Comparison)*. Väčšina takýchto algoritmov je založená na porovnávaní štatistických charakteristík sevencií, ako napríklad metóda frekvencií slov *(angl. Word Frequencies)* [15, 19]:

- Z každej vstupnej sekvencie dĺžky n získaj pre definované k všetky slová = podreťazce dĺžky k. Typicky $k = 8$, $4^8 = 65536$ rôznych slov.
- Zisti počet každého z podreťazcov a vytvor tak vektor frekvencií slov.
- Pri porovnávaní sekvencií nepracuj so znakmi sekvencií, ale s vektormi frekvencií, ktoré sú diskrétne, kratšie a ľahšie manipulovateľné. Použi štandardné algoritmy, napríklad zhlukovú analýzu.

Určitým vylepšením algoritmu je metóda *frekvencií riedkych slov (angl. Spaced Word Frequencies)*, ktorá spočíva v nahradení súvislých oblastí dĺžky k binárnou maskou s väčšou dĺžkou, obsahujúcou indexy pozícií *"match"* a *"do not care"*. Dve slová sa prehlásia za zhodné, ak sa zhodujú na pozíciach označených *"match"*, pričom hodnoty na pozíciach *"do not care"* sa neuvažujú. Výhodou je nižšia vzájomná závislosť dvoch vzo-

riek. Podľa autora algoritmu [18] sú bázy ležiace pri sebe navzájom ovplyvňované a výber vzorky cez väčšie okno zaručí výber nezávislých báz a teda presnejšie vzorkovanie sekvencie.
Iní autori zisťujú počty *di- tri-* a *tetra- nukleotidových početností* podľa sofistikovaných vzorcov [25], čo však nemení štatistickú podstatu metódy.

4.3 VYUŽITIE MATEMATICKÝCH TRANSFORMÁCIÍ NA POROVNÁVANIE REŤAZCOV

Cieľom použitia matematických transformácií na jednorozmerné reťazce (napríklad sekvencie DNA) je (1): Získať ľahšie porovnateľné vektory - diskrétne, číselné a pod., ktoré (2): Lepšie reprezentujú vstupný vektor. Úlohy a ciele vykonávané v procese porovnávania reťazcov, ako aj existujúci výskum v tejto oblasti opisujú nasledujúce časti.

4.3.1 *Transformácia reťazcov na vstupné vektory*

Signálové transformácie (vo všeobecnosti) vyžadujú zápis signálu (dát vstupného vektoru) v číselnej forme. Pre sekvencie DNA, ktoré obsahujú 4 prvky sú typické dva prístupy prepisu báz na číselné hodnoty:

- **Reprezentácia koreňmi jednotky**: Hodnotám báz (A, C, G, T) sa priradia do vstupného vektora kódové hodnoty z množiny komplexných čísel tak, aby tieto boli koreňmi jednotky. Jedno možné kódovanie je: $'A' = 1, 'C' = -i, 'G' = i, 'T' = -1$. Vstupný znakový reťazec GCATAACTTG sa transformuje na vektor [i,-i,1,-1,1,1,-i,-1,-1,i]. Tento postup využíva práce vychádzajú z *MAFFT* [17, 10].
- **Reprezentácia binárnymi indikátormi**: Sekvencia sa reprezentuje c binárnymi vektormi, kde c je kardinalita vstupnej abecedy. Každý reprezentuje jeden znak abecedy, pričom v indikátorovej sekvencii BSI_i znaku i sa nachádza hodnota 1, ak sa v reťazci nachádza znak i na zodpovedajúcej pozícií a hodnota 0 inak. Napríklad reťazec GCATAACTTG sa pretransformuje na vektory A->[0,0,1,0,1,1,0,0,0,0], C->[0,1,0,0,0,0,1,0,0,0] a pod. Túto stratégiu využíva podobne zameraná práca skúmajúca použitie DFT pri úlohe hodnotenia podobnosti sekvencií [31].

Ani jeden z prístupov však nedokáže implicitne korigovať chyby sekvenátorov, v dôsledku ktorých sa v sekvencii vyskytujú časti v reverznom poradí a s komplementárnymi bázami, teda $'T'$ namiesto $'A'$ a $'G'$ namiesto $'C'$ a vice versa. Na ošetrenie menovaného problému navrhujeme prístup založený na kódovaní iba nekomplementárnych báz

- **Bipolárna reprezentácia**: Hodnotám báz sa priradia iba reálne kódové hodnoty, napr. $'A' =' T' = 1, 'C' =' G' = -1$, dbajúc na komplementaritu báz A-T a C-G.

Pri použití tohto typu je v takomto prípade signálové spektrum sekvencie a jej reverzovaného komplementárneho ekvivalentu zhodné.

4.3.2 *Práce iných autorov*

Changchuan Yin: Meranie podobnosti DNA sekvencií Fourierovou transformáciou s využitím v hierarchickom zhlukovaní

Changchuan Yin et al. sa vo svojej pomerne aktuálnej práci [31, 32, 30] zaoberajú využitím FT na zisťovanie vzájomnej (*pairwise*) podobnosti dvojíc sekvencií. Algoritmus *spektrálnej transformácie podľa Yin et al. [31]* je jednoduchý a priamočiary, no napriek tomu, podľa autorov dosahuje dobré výsledky. Postup je nasledovný:

- **Vstup**: Tri sekvencie DNA:
  $\{SEQ1(dĺžkyN1), SEQ2(dĺžkyN2), SEQ3(dĺžkyM)\}$;
  $M \geqslant N1, M \geqslant N2$
- **Výstup**: Tri párové vzdialenosti medzi sekvenciami $\{|SEQ1 - SEQ2|, |SEQ1 - SEQ3|, |SEQ2 - SEQ3|\}$
- **Proces**:
  1. Vytvor zo vstupných vektorov $SEQ1, SEQ2, SEQ3$ binárne sekvencie $BS1, BS2, BS3$ (viď vyššie).
  2. Spočítaj Fourierove spektrá z binárnych reťazcov $BS1 \rightarrow PS1$, $BS2 \rightarrow PS2, BS3 \rightarrow PS3M$.
  3. Rozšír spektrá $PS1(dĺžkyN1)$ a $PS2(dĺžkyN2)$ na dĺžku M. Vzniknú rozšírené $PS1M, PS2M$.
  4. Vypočítaj jednoduchú Euklidovskú párovú vzdialenosť medzi spektrami $PS1M, PS2M, PS3M$:

$d(PS1M, PS2M)$, $d(PS1M, PS3M)$, $d(PS2M, PS3M)$.
Tieto sú výstupom algoritmu.

K práci si dovolíme uviesť niekoľko poznámok:

- Z každého vstupného vektora by mali vzniknúť štyri binárne sekvencie, každá reprezentujúca jeden znak abecedy {A, C, G, T}. V práci sú spomínané tri binárne sekvencie, pre každý vstupný vektor jedna. Popis konkrétnej realizácie mapovania štyroch reťazcov na jeden alebo prípadného zanedbania niektorých znakov v práci chýba.
- Signál reprezentovaný binárnou sekvenciou BSx je diskrétny a nie periodický. Transformácia takéhoto signálu je spojitá [22]. Je teda možné zistiť aj hodnoty frekvenčného spektra, ktoré nereprezentujú frekvenciu $n.\frac{1}{N}; n \in P$. V tomto prípade nulové hodnoty doplnené ku kratšej zo sekvencií tvoria pridanú informáciu potrebnú k dopočítaniu ďalších medziľahlých hodnôt. Vo vyššie popísanom postupe by preto bolo pravdepodobne efektívnejšie nahradiť kroky 2 a 3 nasledovne:
  2. Rozšír binárne sekvencie $BS1(dĺžkyN1)$ a $BS2(dĺžkyN2)$ na dĺžku M doplnením nulami na konci. Vzniknú rozšírené $BS1M, BS2M$.
  3. Spočítaj Fourierove spektrá z binárnych reťazcov $BS1M \rightarrow PS1M, BS2M \rightarrow PS2M, BS3 \rightarrow PS3M$.

Tung Hoang: Nová metóda na zhlukovanie DNA sekvencií s využitím Fourierovho spektra

Metóda navrhnutá Tung Hoang et al. [14] (ďalej *metóda momentového spektra podľa Yin et al. [14]*) je pokračovaním vyššie popísaného postupu. Bližšie sa pritom zaoberá konkrétnou realizáciou porovnania dvoch podreťazcov s využitím v hierarchickom zhlukovaní. Autori za hlavný prínos uvádzajú zavedenie tzv.*momentov* ako parametrov zhlukovania. Podľa definície [14] je $j-$tý moment pre konkrétny znak abecedy $\{A, C, G, T\}$, napríklad A definovaný ako:

$$M_j^A = \alpha_j^A \sum_{k=0}^{N-1} (PS^A(k))^j; j = 1, 2, 3...$$

$PS(k)$ je $k-$tá zložka silového spektra, teda druhá mocnina veľkosti $k-$tej zložky výstupného vektora transformácie, $PS(k) = |X(k)|^2$.

Autori navrhujú aj vzťah na výpočet faktorov α_j^A a konečný moment má tvar

$$M_j^A = \frac{1}{N_A^{j-1} N^{j-1}} \sum_{k=0}^{N-1} (PS^A(k))^j; j = 1, 2, 3...$$

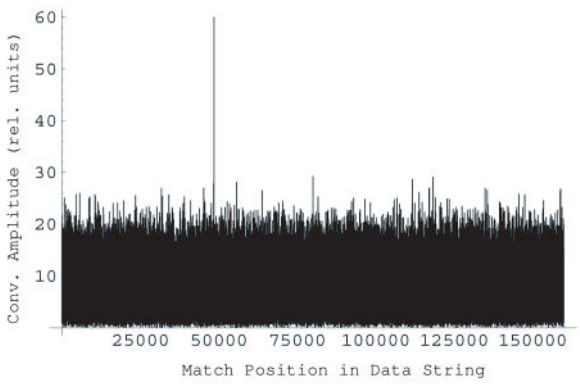
kde N je počet všetkých znakov v sekvencii a N_A počet znakov A v sekvencii. Výpočet sa analogicky prevedie pre všetky znaky abecedy $\{A, C, G, T\}$ a získa sa tým niekoľko momentov
$\{M_1^A, M_1^C, M_1^G, M_1^T, M_2^A, M_2^C, M_2^G, M_2^T, M_3^A, M_3^C, M_3^G, M_3^T\}$. Tieto sa použijú ako charakteristiky v procese zhlukovania. Autori si od procesu sľubujú jednak redukciu dimenzionality - z veľkého vektora charakteristík o dĺžke zodpovedajúcej dĺžke sekvencie sa získa vektor o dĺžke 12. Rovnako sa odstráni problém spracovania sekvencií rôznych dĺžok - vektor má dĺžku 12 pre akúkoľvek sekvenciu. Diskutabilná je výpovedná hodnota takejto 12-tice charakteristík.

Russel W. Hanson: Analýza DNA sekvencií využitím FFT

Russel W. Hanson [10] vo svojej práci využíva spektrá FT aj na problémy zarovnania reťazcov, ktoré, ako bude ukázané neskôr, budeme potrebovať čiastočne poznať aj v našom výskume.
Autor využíva vlastnosť FT, že Fourierova transformácia z konvolúcie dvoch vstupných reťazcov je súčin dvoch Fourierovych transformácií reťazcov samotných. Pojmom súčin sa myslí *bodový (Hadamardov) súčin*, ktorý predstavuje súčin po prvkoch vektora: $\forall i : O_i = I1_i.I2_i$. Vlastnosť je popísaná vzťahom:

$$x \times y = DFT^{-1}(DFT(x) \odot DFT(y))$$

Kde x je hľadaná vzorka a y databáza. Výsledný vektor c indikuje zhodu dĺžky d na pozícií $N - n - 1$, pokiaľ $|c_n|^2 = d$. Autor používa prvý spomínaný prístup transformácie postupností na numerické vektory.



Obr. 6: Výstup analýzy zarovnania sekvencií pomocou FT. [1]

4.3.3 *Vyhodnotenie porovnania podobnosti reťazcov*

Suzana Vinga et al. priniesli podrobný prehľadový článok [28], v ktorom popisujú a porovnávajú metódy vyhodnotenia - kvantifikácie rozdielu v multidimenzionálnom priestore, ktorý tvoria dlhé vektory atribútov - charakteristík. Článok pochádza síce z roku 2002, teda z času najväčšieho rozmachu štatistických metód porovnávania reťazcov, pracuje s frekvenciami krátkych slov a neuvažuje s matematickými transformáciami, no väčšina princípov popísaných v práci je použiteľných aj na transformované dáta.

Váhovaná Euklidovská vzdialenosť

Váhovanie slov má odstrániť nerovnomerný vplyv rôznych slov na výsledok. Ide o úpravu klasickej Euklidovskej vzdialenosti o priradenú váhu slova ρ

$$\mathrm{dist}^2_{WE}(X,Y) = \sum_{i=1}^{N} \rho_i (a_i(X) - a_i(Y))^2$$

1 Obrázok prebratý z [10].

Kovariančné metriky

Úpravou Euklidovskej vzdialenosti o kovariančnú štruktúru vznikne tzv. *Malanahobisova vzdialenosť*:

$$\mathtt{dist}_C(X,Y) = \sum_{i=1}^{N}\sum_{j=1}^{N}(a_i(X) - a_i(Y)).s_{i,j}^{inv}.(a_j(X) - a_j(Y))$$

Matica s je kovariančná matica medzi atribútmi a_i, a_j. Pre veľký počet atribútov je takýto výpočet zjavne časovo náročný.

Metódy založené na uhle v n-dimenzionálnom priestore

Majme dva vstupné vektory o n prvkoch. Pokiaľ tieto vektory určujú bod v n-rozmernej karteziánskej sústave, spojnica počiatku sústavy a takto daného bodu predstavuje definovanú úsečku. Medzi dvomi takýmito úsečkami je možné vypočítať uhol podľa vzťahu:

$$\mathtt{dist}_{ANG}(X,Y) = \frac{\sum_{i=1}^{N} a_i(X).a_i(Y)}{\sqrt{\sum_{i=1}^{N} a_i(X)^2}.\sqrt{\sum_{i=1}^{N} a_i(Y)^2}}$$

Zaujímavou vlastnosťou metódy je jej nenáchylnosť na opakovania, ktoré sa prejavia väčšími hodnotami frekvencií slov, no v nezmenenom pomere. Pripomíname, že článok sa zaoberá metrikami, ktorých vstupom je histrogram slov v sekvenciách.

Časť II

NÁVRH

5

NÁVRH ALGORITMOV POROVNÁVANIA REŤAZCOV

Ako bolo spomenuté, globálnym cieľom výskumnej práce je zlepšiť metódy hierarchického zhlukovania dlhých reťazcov znakov reprezentujúcich reťazce DNA aplikovaním viacerých druhov matematických transformácií z domény spracovania signálu. Nasledujúca kapitola navrhuje postupy použitia týchto transformácií s ohľadom na povahu a vlastnosti dát, ale tiež uvažujúc zamýšľaný výstup, časovú a pamäťovú náročnosť riešenia.

5.1 PRÍSTUPY K POROVNÁVANIU REŤAZCOV

Pri hierarchickom zhlukovaní je potrebné poznať párové vzdialenosti medzi jednotlivými entitami. Tieto je možné získať dvomi spôsobmi:

- Dáta porovnať pomocou špecifických vzdialenostných funkcií, ktoré budú mať na vstupe reťazce DNA. Vzhľadom na povahu práce by mali používať matematické transformácie.
- Transformovať reťazce DNA na numerické vektory, ktoré môžu byť vstupom generických vzdialenostných funkcií, ako napr. Minkowského vzdialenosť. Je vhodné numerickú reprezentáciu získať ako matematickú transformáciu reťazca.

Existujú teda dva možné prístupy, každý s výhodami a nevýhodami.

5.1.1 *Párový prístup založený na transformačných vzdialenostných funkciách*

Tento prístup reflektuje prvý menovaný prístup a teda algoritmy spadajúce do tejto kategórie vytvárajú aplikačne-špecifické vzdialenostné funkcie. Úlohou funkcie je načítať vstupné dáta porovnávaných entít, transformáciou získať frekvenčnú [resp. pri niektorých transformáciách temporálno-frekvenčnú] reprezentáciu dát a na základe nej vyhodnotiť vzájomnú podobnosť. Keďže počítanie transformácie považujeme za

časovo náročný krok, je zrejmé že prístup, ktorý počíta transformácie vektorov vstupných dát pre každú dvojicu elementov má značnú časovú náročnosť, konkrétne: $O(n^2.m.log(m))$ kde n je počet sekvencií a m je (stredná) dĺžka sekvencií. Na druhej strane, tieto metódy majú minimálnu pamäťovú náročnosť, nakoľko nie je potrebné udržiavať v pamäti vektory charakteristík pre všetky sekvencie, ale iba pre dve, ktoré sú porovnávané v čase.

5.1.2 *Lineárny prístup založený na transformačnej filtrácii vstupných dát*

Metódy, ktoré označujeme ako *lineárne filtre* majú za cieľ vykonať transformácie každej vstupnej entity práve raz. Takto sa získa vektor charakteristík reflektujúci spektrálnu reprezentáciu entity, ktorý bude mať typicky menšiu kardinalitu, ako vektor vstupných dát a zároveň kvôli svojej spektrálnej povahe nebude brať do úvahy transpozície vo vstupných dátach. Časová náročnosť bude $O(n.m.log(m))$.

5.1.3 *Zaradenie metód vyžadujúcich zarovnanie*

V kontexte porovnávania vlastností rôznych prístupov uvádzame aj obdobné kategorizovanie metód vyžadujúcich zarovnanie vstupných postupností. Zarovnanie je párová operácia, vyhodnocuje sa teda pre každú dvojicu entít. Časová náročnosť zarovnania je kvadratická, výsledná náročnosť je teda: $O(n^2.m^2)$, resp. pre MUSCLE $O(n^2.m)$ [5]. Oproti tomu sú obidva vyššie spomenuté prístupy menej náročné. Pripomíname však, že v tejto rovine ide o teoretické úvahy o náročnostiach.

Z dôvodu spínanej vyššej časovej náročnosti prístupu párového porovnávania sa metódam vzdialenostných funkcií venujeme vo výrazne menšej miere. Oproti filtrovým metódam disponujú z princípu jednou veľkou výhodou: Dokážu posúdiť kontext dvoch sekvencií použitím výrazne väčšieho počtu charakteristík, ako by bolo možné posudzovať filtrovou implementáciou. Preto sme navrhli jednu metódu aj z tejto skupiny, s cieľom posúdiť jednak kvalitu ostatných navrhovaných metód, ale hlavne vhodnosť transformácií z domény spracovania signálu na bioinformatické dáta ako takú.

5.2.1 *Metóda lokálneho konvolučného zarovnávania sekvencií*

Spomínaná metóda lokálneho zarovnania sekvencií MAFFT [17] alebo aj prístup R. Hansona [10] popísaný v časti 4.3.2 využívajú konvolúciu dlhého reťazca DNA a kratšej vzorky na nájdenie výskytu otočenej vzorky v dlhom reťazci. Pre naše účely potrebujeme zistiť približnú[1] podobnosť dvoch reťazcov a na to nám postačuje náhodne alebo systematicky vybrať k kratších úsekov - vzoriek jednej sekvencie, zistiť a kvantifikovať ich najlepšie zarovnanie v kompletnej druhej sekvencii a z k takto získaných kvantifikátorov určiť výslednú podobnosť. Je nutné podotknúť, že aby sme zachovali jednu zo základných vlastností vzdialenostných funkcií - symetrickosť, je postup potrebné vykonať použijúc obidve sekvencie ako zdroj vzoriek aj ako kompletnú referenčnú sekvenciu a výslednú vzdialenosť určiť ako minimum alebo váhovaný priemer týchto dvoch hodnôt. V opačnom prípade by symetrickosť nebola zachovaná, napríklad, pokiaľ by bola jedna sekvencia podreťazcom druhej.

1 Presnú podobnosť je možné získať pomalými metódami úplného zarovnania sekvencií. Navrhované heuristiky nedokážu ani teoreticky úplne presne kvantifikovať podobnosť reťazcov. Cieľom je, čo najviac sa k presnej hodnote priblížiť využitím čo najmenej výpočtových zdrojov.

Numerická reprezentácia sekvencií pre výpočet konvolúcie

Konvolúcia dvoch signálov [resp. funkcií, reťazcov] zisťuje súčet bodových násobení funkčných hodnôt cez pohyblivé okno. Formálne:

$$(f * g)[n] = \sum_{m=-M}^{M} f(n-m).g(m)$$

Ak chceme kvantifikovať úplnosť zarovnania vzorky na reťazec, potrebujeme nájsť takú reprezentáciu vzorky - funkcie g, aby jej funkčné hodnoty v násobení s funkciou f boli rovné číslu p v prípade zhody znakov reťazcov a (rovnakému) číslu r v prípade nezhody, napríklad:

		A a + bi	C c + di	G e + fi	T g + hi
A	a + bi	1	-0.1	-0.1	-0.1
C	c + di	-0.1	1	-0.1	-0.1
G	e + fi	-0.1	-0.1	1	-0.1
T	g + hi	-0.1	-0.1	-0.1	1

Pre veľkosť abecedy rovnú dvom je návrh reprezentatívnych hodnôt jednoduchý. Použijúc napríklad $A = 1, B = -1$ je výsledok rovný jednej v prípade zhody a mínus jednej inak. Pre štyri znaky je ale možné teoreticky dokázať, že takáto reprezentácia neexistuje, a to ani použijúc komplexné koeficienty. Môžeme jedine navrhnúť koeficienty tak, aby na diagonále boli rovnaké kladné výsledky a mimo diagonály výsledky, ktoré v priemere dávajú nulu a teda by nemali ovplyvňovať výsledok. V implementácií používame nasledovné hodnoty:

		A 1	C 1	G -i	T i
A	1	1	1	0	0
C	1	1	1	0	0
G	i	0	0	1	-1
T	-i	0	0	-1	1

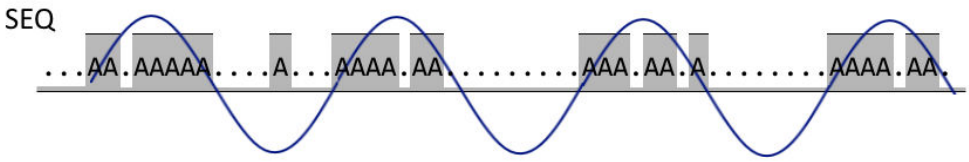
Popis algoritmu

- **Vstup**: Dve sekvencie DNA, veľkosť okna, krok posunutia okna: $\{SEQ1, SEQ2, N_w, N_s\}$
- **Výstup**: Hodnota párovej vzdialenosti dvoch sekvencií.
- **Proces**:
  1. Pre dvojicu sekvencií $SEQ1, SEQ2$:
  2. Zisti numerickú reprezentáciu sekvencie $SEQ1$ a vypočítaj FT $S1 = FT(num(SEQ1))$ tejto reprezentácie
  3. Opakuj kroky 4 až 6 pre každé okno W o dĺžke N_w dané sekvenciou $SEQ2$ a posunutím N_S
  4. Zisti modifikovanú numerickú reprezentáciu otočeného reťazca W a vypočítaj FT $S2 = FT(rnum(W))$ tejto reprezentácie
  5. Vynásob vektory $S1, S2$ do vektora S a vypočítaj inverznú FT $I = FT^{-}1(S)$ tohto vektora
  6. Vo vektore I nájdi maximálnu hodnotu I_{max} - táto predstavuje najlepšie zarovnanie sekvencií W a $SEQ1$. Pre úplné zarovnanie bude hodnota rovná veľkosti okna N_w
  7. Zisti priemer všetkých získaných hodnôt $I_{avg} = avg(I_{max})$. Keďže vyhodnocujeme rozdielnosť sekvencií a nie ich podobnosť, výsledkom môže byť napríklad $\frac{1}{I_{avg}}$
  8. Opakuj kroky 2 až 7 pre dvojicu sekvencií $SEQ2, SEQ1$
  9. Výsledkom je minimum alebo vážený aritmetický priemer dvoch medzivýsledkov z krokov 1 a 8.

Časť venovaná metódam nazvaným ako *Metódy transformačných filtrov* popisuje algoritmy, ktoré jednorázovo transformujú sekvencie DNA na numerické vektory a tieto sú ďalej už porovnávané štandardnými funkciami, napr. Euklidovská vzdialenosť, alebo iné, popísané v časti 4.3.3. Keďže sa práca zaoberá využitím matematických transformácií z domény spracovania signálu, budú tieto numerické vektory vychádzať z frekvenčného spektra vstupnej sekvencie.

5.3.1 *Metóda spektrálnej analýzy cez posúvajúce sa okno*

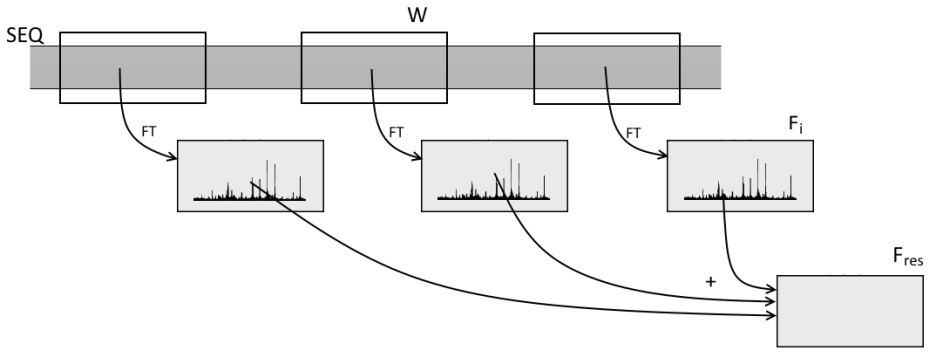
Metóda porovnávania frekvencie slov *Word Frequencies Method* [19], viď časť 4.2.2 pracuje na čisto štatistickom princípe, založenom na pomerne krátkych slovách s typickou dĺžkou 4-8 znakov [19, 31]. Jasnou nevýhodou je malá dĺžka slov, no túto nie je možné zvýšiť z dôvodu veľkej kardinality abecedy (4 znaky pre sekvencie DNA, 20 znakov pre proteínové sekvencie).



Obr. 7: Znázornenie spektrálnej analýzy binárnej sekvencie s oblasťami výskytu indikátora.

O distribúcií znakov naprieč sekvenciou však vypovedá aj frekvenčné spektrum sekvencie. Pokiaľ sa v sekvencii nachádzajú oblasti s vyššou hustotou výskytu sledovaného znaku, v spektrálnej doméne budú dominantné koeficienty, ktoré zodpovedajú perióde vzdialenosti takýchto oblastí. Pre vzorovú sekvenciu na Obr. 7 to bude koeficient pre periódu o dĺžke 8 znakov. Transformácia nájde všetky takéto vzory naprieč sledovanou vzorkou, ktorej dĺžka je rádovo väčšia, ako technický limit dĺžky slov z metódy *Word Frequencies*. Keďže v sekvenciách DNA sa často nachádzajú transpozície, t.j. časti sekvencií sa nemusia ani v príbuzných sekvenciách vyskytovať v rovnakom poradí, chceme stále za-

chovať štatistický charakter metódy. Tento docielime počítaním transformácií cez posúvajúce sa okno a sčítaním všetkých spektrálnych oblastí do jednej.



Obr. 8: Schématické znázornenie algoritmu popísaného v časti 5.3.1

Popis algoritmu

Algoritmus je možné implementovať použitím akejkoľvek spektrálnej transformácie. Pri FFT je možné v *numerickej reprezentácií sekvencie* použiť aj komplexnú reprezentáciu ako v bode 1 časti 4.3.1. Pri transformáciách s celočíselným alebo reálnym vstupom je potrebné používať binárne indikátory báz - jednu pre každý znak abecedy alebo transformáciu počítať iba pre jeden znak abecedy celkovo. Uvádzame prvý menovaný postup:

- **Vstup**: Sekvencia DNA, Veľkosť okna, Krok posunutia okna: $\{SEQ(dĺžkyN), N_w, N_s\}$
- **Výstup**: Vektor spektrálnych charakteristík s dĺžkou: $\{N_w\}$
- **Proces**:
  1. Vytvor vektor nulových hodnôt F_{res} ako výsledok spektrálnej analýzy sekvencie SEQ
  2. Opakuj kroky 3-5 pre každé okno W o dĺžke N_w dané sekvenciou SEQ a posunutím N_s
  3. Vytvor zo vstupného vektora $SEQ[W]$ numerickú reprezentáciu $num(W)$
  4. Spočítaj FFT F_i numerickej reprezentácie vektora $num(W)$.
  5. Sčítaj vektory F_{res} a F_i

6. Vektor F_{res} je po skončení všetkých iterácií výsledkom algoritmu.

Metóda sa osvedčila pracujúc s oknami dĺžky niekoľko tisíc bp, ktoré sa výrazne prekrývali. Podrobnosti o výsledkoch sú uvedené v časti venovanej testovaniu.

5.3.2 *Metóda dominantných spektrálnych koeficientov*

Pri práci s frekvenčnými spektrami získanými zo sekvencií DNA je potrebné si opäť uvedomiť, že sekvencia DNA nie je periodický signál. Je teda pravdepodobné, že frekvenčné spektrum bude obsahovať veľa nenulových koeficientov. Metóda dominantných spektrálnych koeficientov sa snaží vybrať niekoľko najvýraznejších spektrálnych koeficientov a tieto následne poskytnúť vo forme riedkeho vektora <pozícia, hodnota> funkcii, ktorá dokáže takéto vektory porovnať.

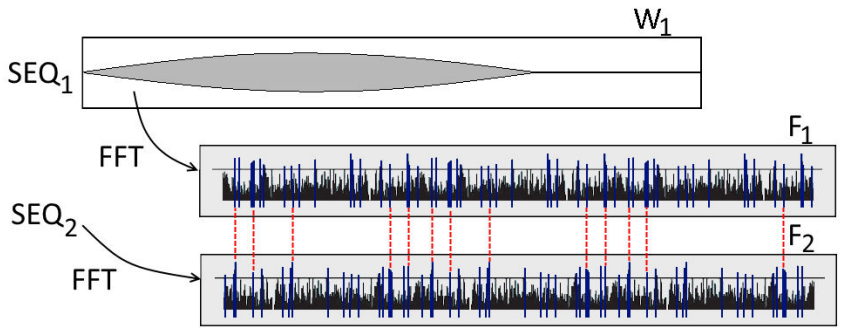
Popis algoritmu transformačnej filtrácie

Algoritmus je opäť možné implementovať použitím akejkoľvek spektrálnej transformácie. Uvádzame spôsob s použitím FFT a komplexnou reprezentáciou znakov abecedy:

- **Vstup**: Sekvencia DNA, Parameter podielu dominantných koeficientov, Max. dĺžka spomedzi všetkých sekvencií:
  $\{SEQ, N_{thr}, N_{max}\}$
- **Výstup**: Riedky vektor dominantných spektrálnych koeficientov.
- **Proces**:
  1. Zisti numerickú reprezentáciu sekvencie SEQ.
  2. Doplň numerickú reprezentáciu sekvencie nulami na dĺžku najdlhšej sekvencie v dátovej množine. Krok je dôležitý z dôvodu správneho škálovania spektier.
  3. Aplikuj zahladzovacie okno (voliteľné).
  4. Vypočítaj FT $S = FT(num(SEQ))$ tejto reprezentácie a zisti absolútne zložky spektrálnych koeficientov. $PS = abs(S)$
  5. Zisti pivotný prvok - $N_{thr}.N_{max}$ -tý najväčší prvok vektora PS.

6. Do výsledku vlož hodnoty a pozície takých prvkov vektora PS, ktorých veľkosť je väčšia, nanajvýš rovná veľkosti pivotného prvku. Takýto riedky vektor je výsledkom transformácie.

Je nutné podotknúť že samotné absolútne hodnoty spektrálnych koeficientov sú irelevantné, keďže podľa *Parsevalovho teorému* budú veľkosti koeficientov priamo úmerné dĺžke sekvencie, z ktorej vznikli. Je ale možná normalizácia reciprokou hodnotou dĺžky sekvencie voči maximálnej dĺžke N_{max}.



Obr. 9: Schématické znázornenie algoritmu popísaného v časti 5.3.2

Popis algoritmu porovnania vektorov

- **Vstup**: Dva riedke vektory dominantných koeficientov: $\{V1, V2\}$
- **Výstup**: Hodnota párovej vzdialenosti dvoch sekvencií.
- **Proces**:
  1. Zisti podobnosť dvoch vektorov ako počet zhôd pozícií dominantných koeficientov N_{match}
  2. Transformuj podobnosť N_{match} na rozdielnosť ako $D = \frac{1}{N_{match}}$ alebo $D = \frac{card(V1)+card(V2)}{N_{match}}$. Keďže normalizácia prebehla doplnením sekvencií na rovnakú dĺžku vo filtračnej transformácii, ďalšia normalizácia už nie je potrebná.
  3. Rozdielnosť D je výsledkom algoritmu.

5.4 METÓDY FILTROV S VYUŽITÍM MRA

5.4.1 *Analýza s viacerými rozlíšeniami: Motivácia*

Mutačné procesy sekvencie DNA sú charakteristické okrem iného inserciami [resp. deléciami] znakov z jednotlivých regiónov, pre podrobnosti viď časť 4.2. Takáto insercia [delécia], hoci aj krátkej časti sekvencie, výrazne ovplyvní spektrálne koeficienty prislúchajúce vysokým frekvenciám na celom sledovanom okne. Z tohto vyplýva potreba sledovať vysoké frekvencie v rámci menších okien, kedy spomínaný jav ovplyvní hodnoty v danom okne, no neovplyvní spektrálne koeficienty v susedných oknách. Prístup je známy v doméne spracovania signálu a označuje sa ako analýza s viacerými rozlíšeniami [3] *angl. Multiple Resulution Analysis - MRA*, d'alej MRA. Pre naše účely budeme používať Haarovu transformáciu, ako špecializáciu Vlnkových transformácií a modifikovanú Fourierovu transformáciu.

5.4.2 *Metóda MRA vlnkových transformácií*

Haarova transformácia, viď časť 3.2.2 je najjednoduchšou transformáciou z množiny Vlnkových transformácií. Používa ortogonálnu vlnkovú funkciu a je teda teoreticky vhodná na spracovanie signálov nadobúdajúcich diskrétne funkčné hodnoty (vrátane dát DNA).
Haarova transformácia vracia zo vstupného vektora dĺžky N výstupný vektor rovnakého dátového typu dĺžky N. Tento obsahuje $\frac{l_{seq}}{2^k}$ koeficientov zodpovedajúce periódam dĺžky 2^k; $k \in < 0, \log_2(l_{seq}) - 1 >$ a jeden údaj o DC komponente. Je zrejmé, že počet prvkov radu konverguje k číslu N. V každom okne sa hodnota vlnkovej funkcie integruje s hodnotami vstupného vektora, ako bolo spomenuté v časti 3.2.2. Získame teda aj temporálnu informáciu o zastúpených frekvenciách, no z dôvodu možných transpozícií, popísaných v predošlej kapitole, je táto irelevantná a je legitímne všetky koeficienty prislúchajúce k jednej frekvencii sčítať do jedného. Takto sa získa výstupný vektor dĺžky $\log(N)$, kde N je dĺžka vstupného vektora, kde jednotlivé zložky reprezentujú periódy signálu s dĺžkami 2^p, kde $p = 1..\log(N)$.

Popis algoritmu

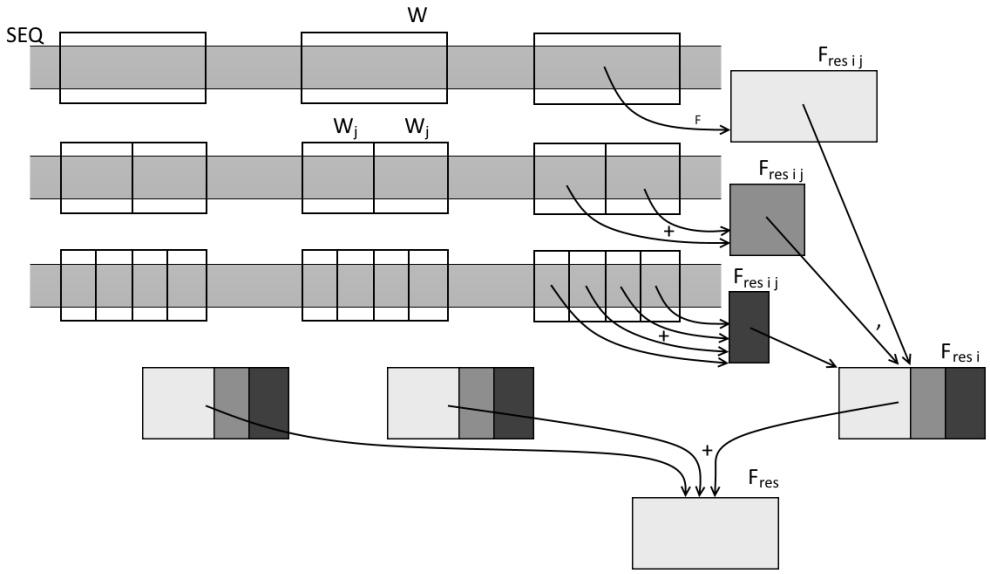
- **Vstup**: Sekvencia DNA, resp. ľubovoľná sekvencia s vopred definovanou kardinalitou použitej abecedy, veľkosť okna, krok posunutia okna:
  $\{SEQ(dĺžkyN), N_w, N_s\}$
- **Výstup**: Vektor spektrálnych charakteristík pre každý sledovaný znak abecedy s dĺžkou:
  $\{log(N_w).C\}$, kde C je kardinalita sledovanej abecedy, najčastejšie $C = 4$.
- **Proces**:
  1. Opakuj body 2-7 pre každý znak abecedy:
  2. Vytvor vektor nulových hodnôt H_{ires} ako výsledok spektrálnej analýzy znaku i
  3. Opakuj body 4-7 pre každé okno W o dĺžke N_w dané sekvenciou SEQ a posunutím N_S
  4. Vytvor zo vstupného vektora $SEQ[W]$ binárnu sekvenciu BS_i, kde i je sledovaný znak.
  5. Spočítaj Haarovu transformáciu H_i vektora BS_i.
  6. Sčítaj koeficienty zodpovedajúce rovnakej frekvencii do redukovaného vektora H_{ired}
  7. Sčítaj vektory H_{ires} a H_{ired}
  8. Spoj vektory H_{ires} do jedného výstupného H_{res}, tento je výsledkom algoritmu.

Metóda úspešne rieši problém príliš veľkých okien pre vysoké frekvencie. Jej problémom ale je, že poskytuje iba spektrálne koeficienty pre periódy s dĺžkou rovnou mocnine dvoch a teda malá veľkosť výsledného indikátorového vektora.

5.4.3 *Metóda MRA spektrálnej analýzy cez posúvajúce sa okno*

Hlavnou nevýhodou predošlej metódy je fakt, že počíta iba spektrálne koeficienty pre frekvencie zodpovedajúce periódam s dĺžkou rovnou mocnine dvoch. Modifikácia metódy počíta FFT pre každú veľkosť okna rovnú mocnine dvoch medzi danou minimálnou a maximálnou veľkosťou. Výsledky získané analýzou s použitím okien menšej veľkosti sa, z vyššie uvedených dôvodov, spočítavajú do jedného vektora.

Uvedený postup počíta s reprezentáciou vstupného vektora komplexnými koeficientami ako v bode 1 časti 4.3.1. Metódu je možné použiť aj pre reprezentáciu binárnymi indikátormi, vtedy sa postup opakuje pre každý znak abecedy a na záver sa vektory zreťazia ako v bode 8 predošlého postupu. Pre jednoduchosť uvádzame prvý spôsob.



Obr. 10: Schématické znázornenie algoritmu popísaného v časti 5.4.3

Popis algoritmu

- **Vstup**: Sekvencia DNA, resp. ľubovoľná sekvencia s vopred definovanou kardinalitou použitej abecedy, max. veľkosť okna, min. veľkosť rozlíšenia, krok posunutia okna:
  $\{SEQ(dĺžkyN), N_{wmax}, N_{wmin}, N_s\}$
- **Výstup**: Vektor spektrálnych charakteristík s dĺžkou N_{wmax}
- **Proces**:
  1. Vytvor vektor nulových hodnôt F_{res} ako výsledok spektrálnej analýzy vstupného vektora.
  2. Opakuj body 3-11 pre každé okno W o dĺžke N_{wmax} dané sekvenciou SEQ a posunutím N_S
  3. Vytvor vektor nulových hodnôt F_{resi} dĺžky N_{wmax} ako výsledok spektrálnej analýzy okna W.
  4. Opakuj body 4-10 pre každú veľkosť pod-okna N_{wj}

5. Vytvor vektor nulových hodnôt F_{resij} dĺžky N_{wj} ako výsledok spektrálnej analýzy okna W pod-oknom dĺžky N_{wj}.
6. Opakuj body 5-9 pre každé pod-okno W_j o dĺžke N_{wj} dané oknom W a posunutím o N_{wj}
7. Vytvor zo vstupného vektora $W[W_j]$ reprezentačnú sekvenciu RS (korene jednotky, binárne indikátory a pod.)
8. Spočítaj FFT F vektora RS.
9. Sčítaj vektory F_{resij} a F
10. Vlož vektor F_{resij} na zodpovedajúce[2] miesto vo vektore F_{resi}
11. Sčítaj vektory F_{resi} a F_{res}
12. F_{res} je výstupom algoritmu.

Situáciu sa pokúša ozrejmiť schéma na Obrázku 10. Metóda má podobné črty ako prístup založený na ST-FFT prístupe, na rozdiel od neho dokáže pracovať aj s dátami obsahujúcimi inzercie a delécie, za cenu prijateľného zvýšenia výpočtovej náročnosti. Metóda sa, podobne ako obdobná metóda bez využitia MRA analýzy, osvedčila pracujúc o oknami dĺžky niekoľko tisíc bp, ktoré sa výrazne prekrývali. Podrobnosti o výsledkoch sú uvedené v časti venovanej testovaniu.

2 Zodpovedajúcim miestom sa myslí miesto, ktoré prislúcha frekvenčnému spektru počítanému daným oknom a žiadnym iným menším oknom. Pre veľkosť okna $N_{wmax} = 2048$ a $N_{wj} = 64$ sa výsledok vloží na pozície 1984..2047, $N_{wj} = 256$ sa vloží na pozície 1792..1919, keďže pozície počínajúc 1920 vyhodnocujú, s väčšou presnosťou, okná s dĺžkou 128.

Časť III

IMPLEMENTÁCIA

6

ELKI: POPIS PLATFORMY ANALÝZY DÁT

ELKI, celým názvom (Environment for deveLoping Knowledge discovery in databases applications supported by Index structures) [1] je open-source implementácia nástrojov pre dátovú analytiku vyvíjaná na Univerzite L'udovíta-Maximiliána v Mníchove. Kostra architektonického riešenia je navrhnutá modulárnym spôsobom. Týmto sa, na rozdiel od obdobných nástrojov ako *Weka*[2] alebo *KNIME*[3][4], umožní jednoduchá doimplementácia komponentov bez potreby rozptýlených zmien a narušenia celkových riadiacich štruktúr, čo je problém v angličtine známy ako *Shotgun surgery*. Za týmto účelom sa v hojnej miere používajú *abstraktné továrne* a pokročilé objektovo-orientované mechanizmy ako napr. *reflexia*. O spôsoboch realizácie bližšie pojednávame v časti 6.2.

6.1 ZÁKLADNÁ ŠTRUKTÚRA APLIKÁCIE

Nástroj ELKI je navrhnutý pre prúdové spracovanie dát. Úloha pozostáva z načítania vstupných dát, výkonu algoritmov, vyhodnotenia výsledku a jeho zápisu. Pre naše účely sú zaujímavé tieto:

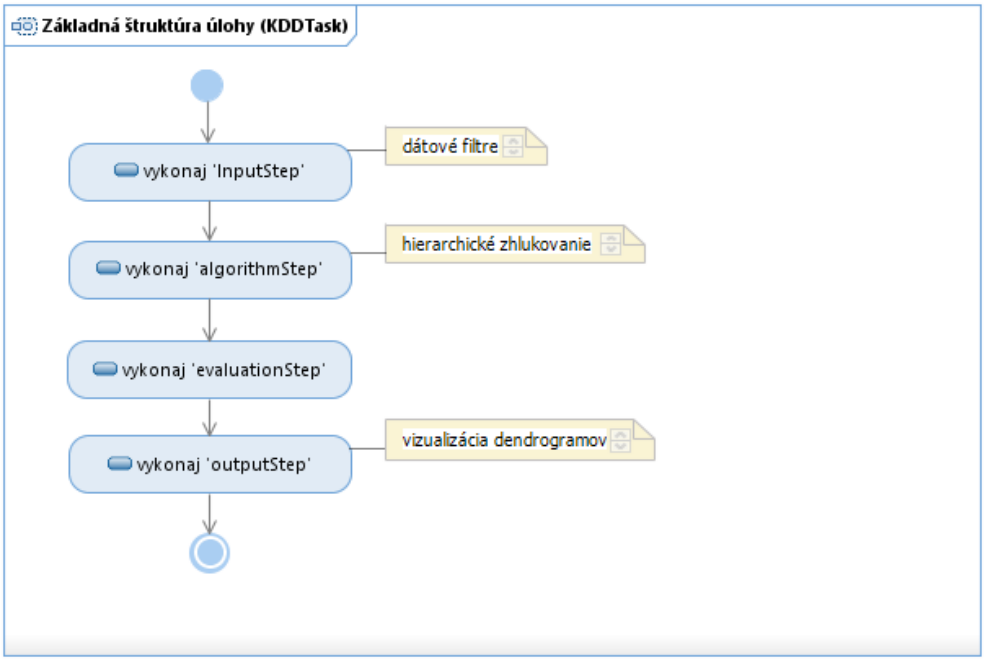
- Vstupný krok, v ktorom sa po, resp. počas čítania vstupných dát vykonáva ich filtrovanie. Pri tom sa kolekcie dát nahradia alebo doplnia novými, ktoré vznikli transformáciou pôvodných.
- Algoritmický krok, kedy sa vykoná samotné zhlukovanie modifikovanými algoritmami hierarchického zhlukovania použijúc navrhnuté vzdialenostné funkcie.
- Výstupný krok, kedy modifikované výstupné triedy zapíšu výsledok zhlukovania a vykreslia dendrogramy.

1 Hlavná stránka nástroja ELKI: https://elki-project.github.io/

2 Open-source Java implementácia nástroja pre strojové učenie a dátovú analytiku. http://www.cs.waikato.ac.nz/ ml/weka/

3 Konstanz Information Miner. Open-source neakademický nástroj dátovej analytiky. http://www.knime.org/

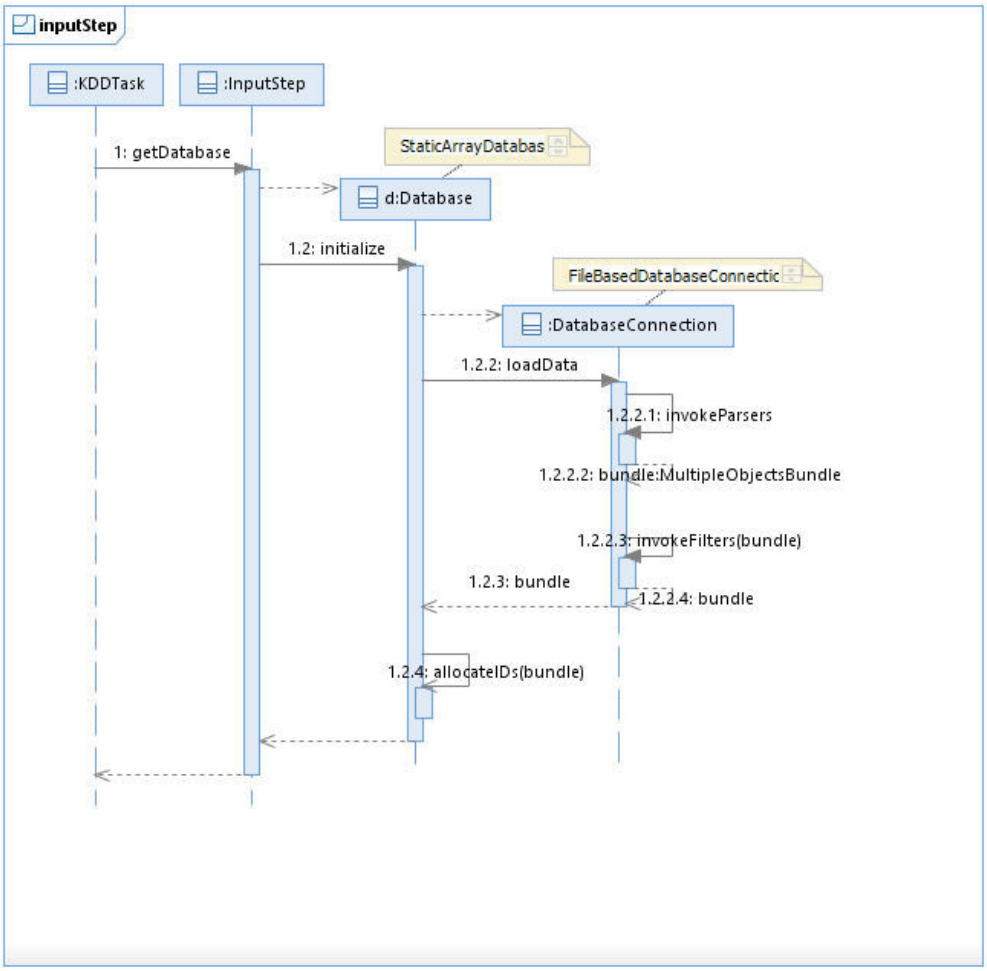
4 Obidva nástroje sú rozsahom výrazne menšie ako nástroj ELKI



Obr. 11: Diagram aktivít základnej štruktúry úlohy.

6.1.1 *Vstupný krok*

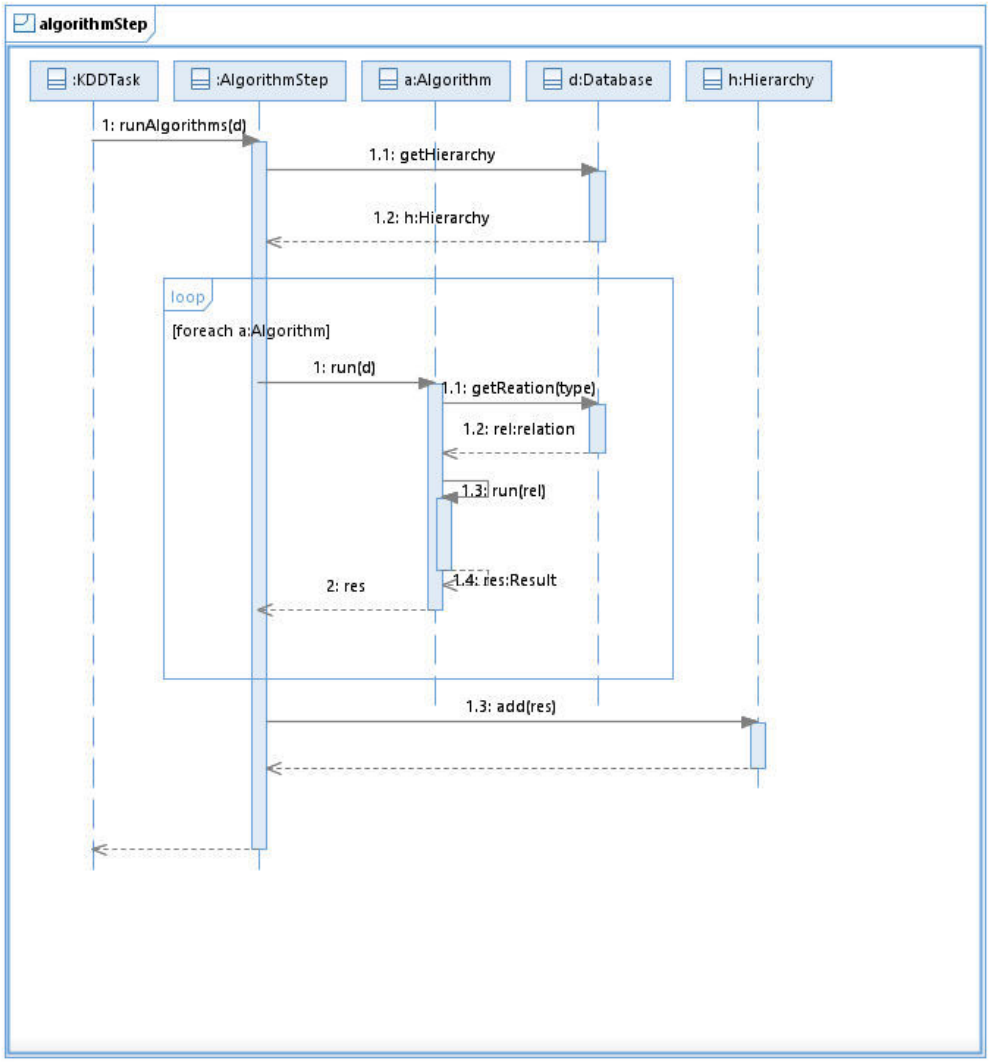
Vstupný krok úlohy vytvorí databázu zo špecifikovaných vstupných dát. Databázu predstavuje kolekcia vektorov rôznych dátových typov podľa zvolených továrenských metód v konfigurácii. Jedným z vektorov je vždy vektor alokovaných identifikátorov. Vykonávané kroky sa menia podľa zvolenej konfigurácie. Diagram na Obr. 12 zobrazuje typickú situáciu s čítaním zo súboru.



Obr. 12: Sekvenčný diagram interakcie počas *inputStep*.

6.1.2 *Algoritmický krok*

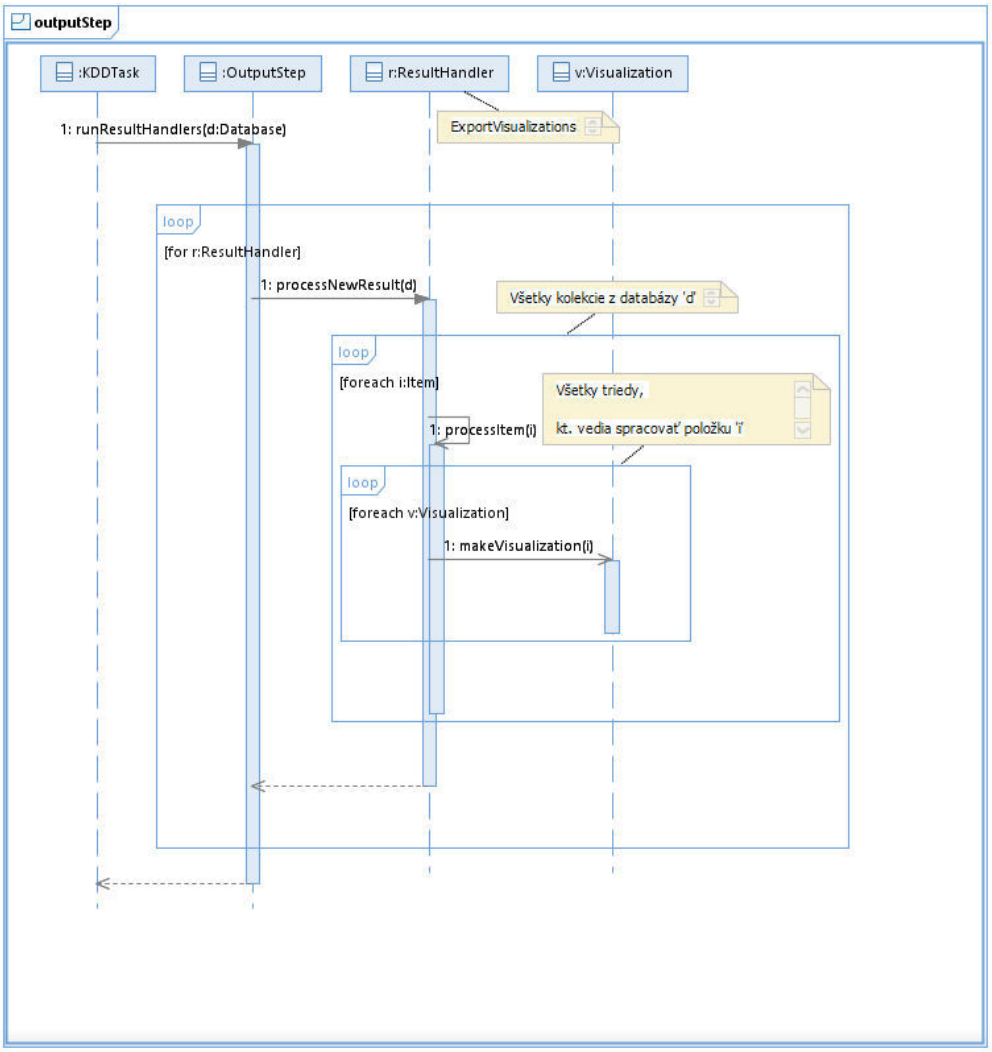
Algoritmický krok pozostáva zo spustenia všetkých konfigurovaných algoritmov, ktoré transformujú vstupné dáta na výsledok. Pri svojom spustení získavajú z kolekcií vstupných dát podľa dátového typu tie, ktoré sú relevantné pre ich spustenie. Následne sa reflexívne nájde a vykoná implementácia metódy *run* taká, ktorá spracúva nájdené kolekcie. Je teda možné preťažovať implementácie podľa nájdených dostupných vstupných dát. Nevýhodou riešenia je viazanie algoritmov na typ dát a teda nemožnosť naviazať konkrétnu inštanciu vektora vstupných dát na konkrétny algoritmus, pokiaľ je k dispozícii viacero vektorov rovnakého dátového typu. Problém budeme rozoberať v ďalšej časti.



Obr. 13: Sekvenčný diagram interakcie počas *algorithmStep*.

6.1.3 *Výstupný krok*

Výstupný krok je podobná transformácia ako algoritmický krok, v tomto prípade sa kolekcia obsahujúca výsledok (inštancia triedy odvodenej od *Result*) transformuje na výstup, textový alebo grafický. Krok prebieha podobne ako predošlý: Každý špecifikovaný *spracovateľ výsledku* (napr. exportér vizualizácií, textový zapisovač výsledku a pod.) prejde všetkými výsledkami a spustí služby, ktoré ich dokážu spracovať. Tieto služby vrátia grafický alebo textový výstup, ktorý spracovateľ zapíše, resp. zobrazí podľa stratégie, ktorú vykonáva.



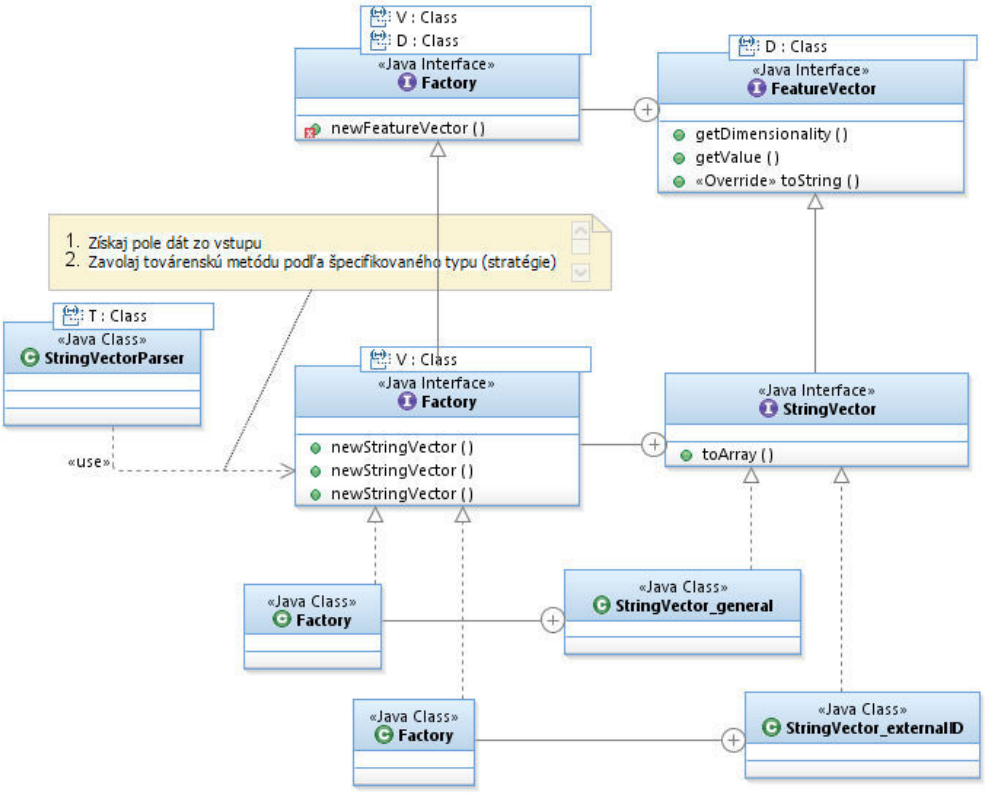
Obr. 14: Sekvenčný diagram interakcie počas *outputStep*.

6.2 ZAVEDENIE ROZŠÍRITEĽNOSTI V ELKI

Za účelom pridávania funkcionality bez potreby zmien v kóde používa ELKI mechanizmy vytvárania inštancií pomocou reflexie cez továrenské metódy.
Realizované sú dva prístupy s podobným princípom.

- Na vytváranie dátových entít sa používajú továrenské metódy z triedy *Factory*, ktorá je referencovaná ako *stratégia* v objekte vytvárajúcom tieto dátové objekty.
- Vytváranie výkonných objektov sa realizuje obdobne, trieda zastrešujúca továrenské metódy je označená ako *Parametrizer*. V tomto

prípade je kreácia obohatená o získavanie jednoduchých parametrov a vykonáva sa viacstupňovo, keďže aj agregovaný objekt môže požadovať parametrizáciu.



Obr. 15: Diagram tried modelu továrenských metód na vytváranie dátových entít.

6.3 KRITIKA NÁSTROJA A POPIS VŠEOBECNÝCH ROZŠÍRENÍ

Nástroj ELKI ešte nemá za sebou oficiálne vydanie[5], je preto pochopiteľné, že sme identifikovali niekoľko jeho nedokonalostí. V tejto časti zhrnieme, vrátane popisu riešenia, tie, ktoré nesúvisia priamo s doménou bioinformatiky. Tieto rozšírenia typicky nevyužívajú mechanizmy rozšíriteľnosti platformy ELKI a, keďže často zasahujú do jadra, menia aj pôvodnú implementáciu.

5 Vývoj ELKI je v súčasnosti (3.4.2017) na verzii 0.7.1.

6.3.1 *Realizované všeobecné rozšírenia*

Nové dátové typy vstupov

Pre potrebu reprezentovať množinu bioinformatických dát externým odkazom sme zaviedli nový všeobecný typ *StringVector*, ktorý obsahuje pole textových reťazcov. Ďalej je od neho odvodený *StringVector_externalID*, popisujúci externú entitu ako generický súbor parametrov, ktoré dokáže spracovať vstupno-výstupná služba špecifikovaná v konfigurácii. Každému dátovému typu zodpovedá novo vytvorená továrenské trieda.
Novovytvorený *StringVectorParser* je parser horeuvedených typov, značne využívajúci znovupoužiteľnosť abstraktných parserov.

Nové dátové typy výstupov

Popri vysoko špecializovaných typoch výsledkov, napr. *PointerHierarchyRepresentationResult* alebo *ConfusionMatrixEvaluationResult* sme zaviedli generické typy výsledkov:

- *StringResult*, ktorý obaľuje textový reťazec a značkuje ho ako výstup algoritmov. Mienený je najmä na výstupnú reprezentáciu štruktúrovaných textových dát (JSON, XML), viď ďalej integráciu so skriptovacími jazykmi.
- *MatrixResult*, ktorý obdobne obaľuje maticu a vektory stĺpcových a riadkových hlavičiek.

K obidvom dátovým typom existujú aj vstupno-výstupné triedy, ktoré zabezpečia serializáciu alebo deserializáciu typov do ľudsky čitateľnej textovej formy.

Všeobecné algoritmy

Do kategórie nových všeobecných algoritmov vykonávaných v rámci algoritmického kroku patria nasledovné:

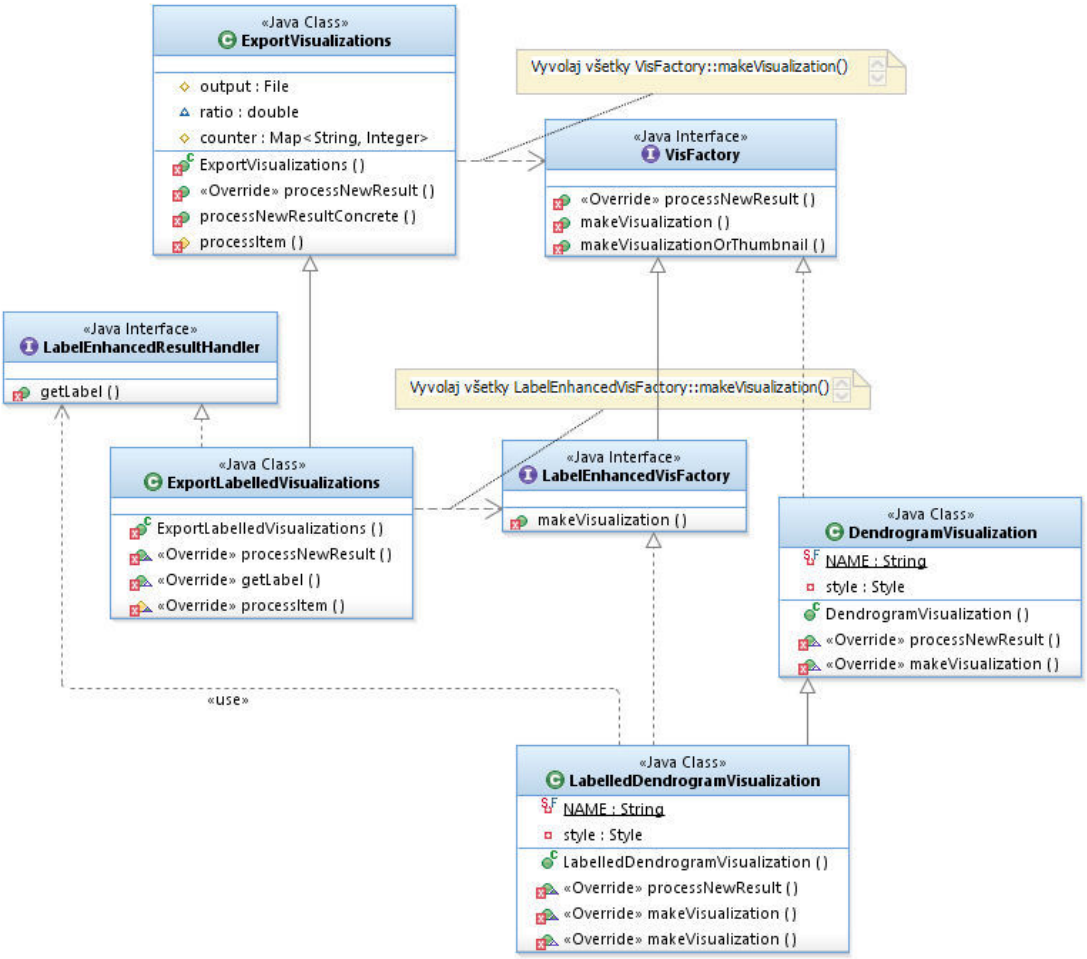
- *Interpreter jazyka JavaScript*. Tento bol zavedený, aby sa umožnilo vytvárať jednoduché nové algoritmické úlohy, bez potreby kompilovania celého nástroja, resp. bez potreby mať k dispozícií zdrojové kódy vôbec. Používa sa pritom výkonný interpreter *Nashorn*

z distribúcie Java8, ktorý je zaintegrovaný do nástroja ELKI štandardným spôsobom rozšíriteľnosti. Výstupom je štruktúrovaný text vo formáte JSON obalený do objektu typu *StringResult*.

- *Algoritmus na tvorbu vzdialenostných matíc*. Jednoduchý algoritmus vypočíta párové vzdialenosti všetkých dátových entít zvolenou metrikou a zapíše ich do nového objektu typu *MatrixResult*. Iniciatíva súvisí s hlavnou úlohou diplomovej práce, no algoritmus je navrhnutý na všeobecné použitie.
- *Nový spôsob práce s externe zapísanými vzdialenosťami entít*. V súčasnosti existuje v ELKI podpora načítavania párových vzdialeností entít z externého súboru, no realizácia sa opiera o silný predpoklad, že dáta sú v matici sú zapísané v počte a poradí rovnakom, ako sú alokované ich ID v programe. Naša realizácia nahrádza existujúcich sedem funkcií jednou, ktorá páruje prvky podľa porovnania hlavičky entít v matici s hlavičkami entít v programe. Sleduje sa pritom aj neúplná zhoda.

Úprava a doplnenie hlavičiek výstupov

V rámci opráv chýb pôvodnej implementácie sme identifikovali problém práce s hlavičkami dátových entít. Pôvodná implementácia ELKI obsahovala triedy na ukladanie hlavičiek (napr. *LabelList*), no tieto dáta sa ďalej nevyužívali a vo výstupoch, či už textových alebo grafických sa entity vyskytovali bez popisov. Takýmto príkladom je dendrogram bez hlavičiek entít, ktorý úplne stráca informatívnosť.
Ako súčasť riešenia sme zaviedli rozhranie *LabelProvider*, ktoré definuje služby priraďujúce hlavičku podľa databázového ID. Zatiaľ jediná implementácia *RelationLabelProvider* používa existujúcu reláciu typu *LabelList*. Časovo náročnejší implementačný krok požaduje doplnenie práce s týmto rozhraním do každej triedy požadujúcej tieto hlavičky, napr. vizualizačné triedy *AutomaticVisualization, ExportVisualizations, ...*, od ktorých je potrebné odvodiť novú verziu so schopnosťou používať tieto hlavičky. Pokiaľ chceme zabrániť duplikácii kódu, napríklad použitím vzoru *Template Method*, je potrebné preorganizovať aj kód pôvodných tried a vyčleniť hlavnú logiku (napr. export dendrogramu do formátu svg) do šablónových metód.



Obr. 16: Diagram tried riešenia zobrazovania hlavičiek pri vykresľovaní dendrogramu.

6.3.2 *Identifikované ďalšie problémy*

Identifikovali sme aj ďalšie potenciálne problémy, no ich vyriešenie je jednak omnoho náročnejšie a nie je pre nás v túto chvíľu potrené. Nimi sú:

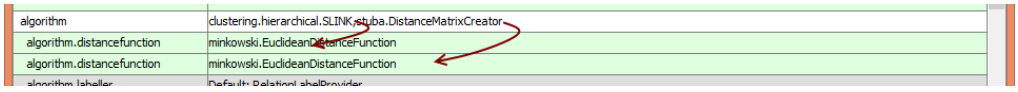
- Nemožnosť vybrať používateľom, ktoré vizualizačné služby sa použijú. V súčasnosti sa použijú všetky služby, ktoré dokážu výsledok spracovať, viď poznámku "Vyvolaj všetky služby triedy Visfactory" v predošlom diagrame. Nejde pritom iba o služby vizualizácie dát. Problém zjavne identifikovali aj samotní autori, keďže v kóde je príslušná poznámka[6].

6 *TODO: allow selecting individual projections only.* V *ExportVisualizations.java*, riadok 145

- Nemožnosť vybrať, ktorá inštancia dátového typu zo vstupného kroku sa použije ako vstup do algoritmického kroku. V súčasnosti sa viažu vstupy na základe typov, (napr. vzdialenostná funkcia *SparseVectorComparable* definuje, že pracuje s typom *SparseNumebrVector*), no neexistuje spôsob ako vyberať medzi viacerými inštanciami relácie rovnakého typu.

Riešenie týchto problémov by mohlo viesť cez vývoj simulovaného prostredia, ktoré by v prvom kroku spustilo podľa konfigurácie falošné služby bez dát, čím by sa zistilo, aké artefakty počas behu programu vznikajú. Následne by používateľ vybral, ktoré inštancie sa majú použiť. Ako bolo spomenuté vyššie, vyriešenie problému pre nás v túto chvíľu nie je potrebné.

- Ďalší problém súvisí s nejednoznačnou a ťažkopádnou prácou s ID dátových entít. Existuje dvadsať rozhraní špecifikujúcich ID dátových entitít a ďalších minimálne osem rozhraní pre prácu s nimi. K tomu existuje rovnaké množstvo implementačných tried. Ide pritom mnohokrát o zbytočné a ťažkopádne obaľujúce triedy, napr. *ArrayDBIDs*, ktorá špecifikuje pole ID entít. Podľa nášho názoru je potrebné zaviesť nové, jednoznačné identifikátory a súčasné ponechať z dôvodu spätnej kompatibility, ale označiť ich za zastarané. Riešenie aj tak vyžaduje značný zásah do jadra platformy.
- Ostatný problém súvisí s architektonickým riešením platformy. Konfigurácie niektorých súčastí môžu zdielať spoločné komponenty. V súčasnom stave sa však všetky závislé súčasti vytvárajú ako *prototypy*, čo má za následok (I.) Zvýšenie pamäťových nárokov, ale najmä (II.) Zvýšenie časovej náročnosti, keďže niektoré výpočty sa realizujú viackrát. Záležitosť je možné riešiť vhodným použitím vzoru *Dependency injection*, resp. prostriedkami komponentovo - orientovaného programovania.



Obr. 17: Duplicitné konfigurácie súčastí zrejmé v používateľskom rozhraní aplikácie.

7

IMPLEMENTÁCIA DOMÉNOVO-ŠPECIFICKÝCH SÚČASTÍ

Doménovo-špecifické súčasti, teda algoritmy riešiace primárnu úlohu spadajú pod nasledovné skupiny:

- Filtre s využitím matematických transformácií popísané v časti *Návrh*.
- Vzdialenostné funkcie s využitím matematických transformácií popísané v časti *Návrh*, pomocné meta-vzdialenostné funkcie (agregácia výsledkov).
- Implementácia referenčných metód podľa dostupnej literatúry, prednostne formou filtra.
- Algoritmy hierarchického zhlukovania (*Neighbour Joining*), pomocné entity na uchovanie, import/export výsledkov.
- Pomocné nástroje pre úlohu spracovania dát (import, parsovanie vstupných dát)
- Pomocné nástroje pre úlohu vyhodnotenia výsledku.

Algoritmy do maximálnej miery využívajú mechanizmy rozšíriteľnosti platformy ELKI. Popis netriviálnych súčastí bude uvedený v nasledujúcich častiach.

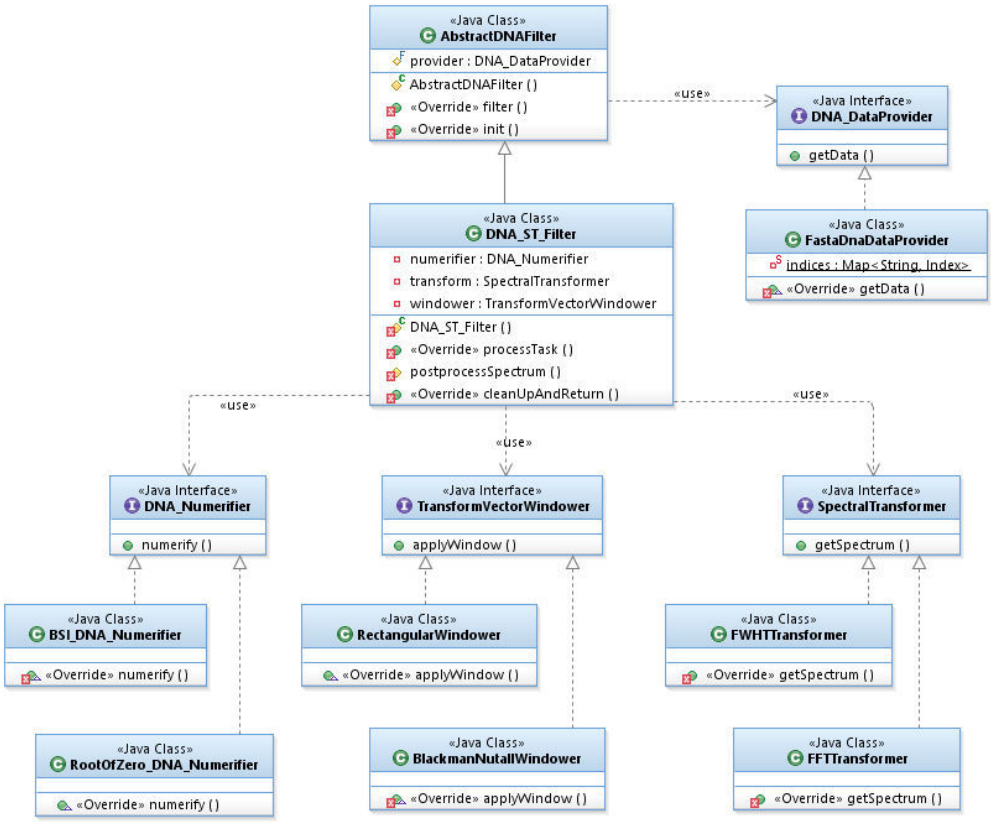
7.1 ARCHITEKTÚRA A POPIS TRANSFORMAČNÝCH FILTROV

Do prvej a tretej skupiny algoritmov patria filtre transformujúce vstupné dáta na krátke numerické vektory určené na porovnanie generickými vzdialenostnými funkciami. V obidvoch prípadoch dodržujú rovnaký vzor:

- Rodičovská trieda *AbstractDNAFilter -> ObjectFilter* definuje schému procesu spracovania dát a predpisuje špecifické metódy určené na prekonanie implementujúcim triedam. Definuje tiež stratégiu číta-

nia dát - parser *.fasta* súborov alebo náhodný generátor sekvencií pre účely ladenia.

- Implementujúce triedy definujú spôsob transformácie dát. V úplnom prípade podobne využívajú schému so vzormi *Strategy* a *Null object*.



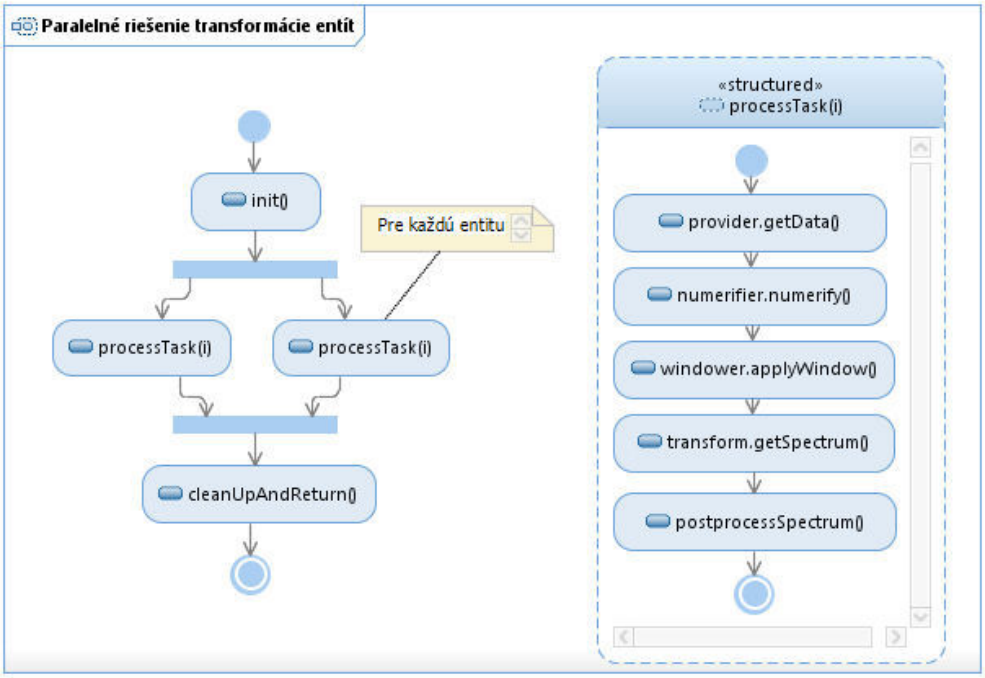
Obr. 18: Diagram tried zúčastňujúcich sa prúdu spracovania sekvencie DNA.

Proces spracovania entity sa skladá z nasledovných krokov:

1. Zvolená stratégia čítania dát na základe meta-dát načítaných zo vstupného súboru poskytne dáta - sekvencie DNA. Tieto sú stále vo formáte znakového reťazca.
2. Znakový reťazec sa zvolenou stratégiou transformuje na numerický vektor, ktorý vstúpi do signálovej transformácie.
3. Vstupný numerický vektor sa zahladí zvoleným oknom. Krok je možné obísť voľbou obdĺžnikového okna.
4. Upravený vstupný numerický vektor sa transformuje do spektrálnej oblasti zvolenou stratégiou spektrálnej transformácie.

5. Z výsledného spektra sa počítajú odvodené numerické charakteristiky.

Nie všetky filtre využívajú všetky popísané kroky, resp. viaceré môžu byť zlúčené do jedného explicitného. Extrakcia metód na inicializáciu filtra a sumarizáciu výsledkov od akcie na spracovanie konkrétnej entity umožní sekvenčný aj paralelný beh filtra podľa diagramu na Obr. 19.



Obr. 19: Diagram aktivít pri paralelnej filtračnej transformácii entít.

7.2 POPIS ŠPECIFICKÝCH VZDIALENOSTNÝCH FUNKCIÍ

Z dôvodu očakávanej vysokej časovej zložitosti bola navrhnutá jediná transformačná vzdialenostná funkcia založená na *lokálnom konvolučnom zarovnávaní sekvencií*.
Implementácia dedí od *AbstractDnaFunction -> AbstractPrimitiveDistanceFunction<StringVector_externalID>* ktorá podobne, ako u transformačných filtrov, predpisuje spôsob napojenia na dáta. Zvyšný proces v implementujúcej triede je špecifický. Použitý spôsob transformácie sekvencií na numerické vektory je pevne daný problémami diskutovanými v časti 5.2.1. Podobne výpočet transformácií (doprednej a spätnej Fourierovej) je špecifický.

Implementovaná bola aj vzdialenostná funkcia založená na práci Yin et. al. [31], no princíp jej fungovania je možné nahradiť oveľa časovo efektívnejším transformačným filtrom, ktorý je ďalej používaný v testoch.

7.3 implementácia pomocných vzdialenostných funkcií

Externé vzdialenostné funkcie *FileBasedMatrixDistanceFunction -> AbstractPrimitiveDistanceFunction<LabelList>*, resp. *AggregativeFileBasedMatrixDistanceFunction -> FileBasedMatrixDistanceFunction* slúžia na načítavanie vzdialeností entít z predpočítaných, externe zapísaných matíc.
V obidvoch prípadoch sa pri inicializácií vybuduje 2D mapa uchovávajúca hodnotu vzdialeností medzi entitami, pričom sa ako referencia používa popis entity - *Label*. Keďže sa testuje na čiastočnú zhodu, algoritmus nepracuje správne, pokiaľ je popis jednej entity podreťazcom popisu druhej.

Časť IV

ZHODNOTENIE

8

TESTOVACIE DÁTA A METRIKY

8.1 POPIS DÁT POUŽITÝCH V EXPERIMENTOCH

Prvotné výsledky získané nesystematickým testovaním ukazujú na viaceré dimenzie variability dátových množín: (I.) Dĺžka sekvencií DNA, ktoré vstupujú do transformačného filtra, resp. vzdialenostnej funkcie. (II.) Stredná podobnosť biologických druhov, ktoré tieto sekvencie popisujú. Inak povedané, niektoré metódy môžu byť lepšie na hrubé rozlíšenie, resp. kategorizovanie do skupín organizmov, iné zase na jemné rozlíšenie v rámci danej skupiny. Paradoxne, tieto môžu zlyhať pri prvej menovanej úlohe, nakoľko sekvencie sú príliš rozdielne. (III.) Pôvod sekvencií. Mitochondriálne sekvencie sú výrazne kratšie a cyklické, preto sa u rozličných druhov môžu vyskytovať s posunutým začiatkom. Jadrové sekvencie sú dlhšie a oproti mitochondriálnym sekvenciám obsahujú transpozície častí genómu s oveľa väčšou pravdepodobnosťou. Je pochopiteľné, že testami máme záujem pokryť maximálny počet možných dátových množín.
Ku všetkým súborom dátových entít vyžadujeme, aby bol k dispozícií overený výsledok. Tento je získaný presnými, no pomalými metódami porovnávania sekvencií vyžadujúcimi zarovnanie. V krajnom prípade dokážeme referenčné podobnosti získať aj vlastným použitím algoritmov vyžadujúcich zarovnanie, v našom prípade algoritmom *MUSCLE* [5]. Vzhľadom na jeho vysokú pamäťovú a časovú náročnosť sme používali vzdialené rozhranie poskytované *EMBL*[1]. Algroritmus poskytne zarovnanie sekvencií (z ktorého je zrejmá stredná podobnosť druhov) a dendrogram vrátane metrických ohodnotení dĺžky hrán.

Používame nasledovné dátové sady:

- Cicavce z prác Yin et. al. [31, 32, 14]. Ide o mitochondriálne sekvencie 31 geneticky podobných druhov cicavcov s dĺžkou 16kbp

1 *The European Molecular Biology Laboratory.* Hlavná stránka algoritmu MUSCLE: http://www.ebi.ac.uk/Tools/msa/muscle/

per sekvenciu.[2] Na získanie referenčného dednrogramu sme použili vlastný beh algoritmu MUSCLE. Na datasete bolo vykonané exhaustívne systematické testovanie, najúspešnejšie konfigurácie boli v ďalších troch iteráciách optimalizované.

- Kvasinky I. z práce Valach et.al. [26] Dátová sada obsahuje súbory s mitochondriálnymi sekvenciami kvasiniek, 8 sekvencii s približnou dĺžkou 1kbp per entitu. Z celkového počtu 39 entít sme vybrali 11 entít patriacich do rôznych podstromov tak, aby boli medzi nimi maximálne vzdialenosti. Práca obsahuje referenčný dendrogram s ohodnotenými dĺžkami hrán.
- Kvasinky II. Dátová sada obsahujúca iný výber 10 druhov z predošlej množiny tak, aby tvorili spoločný podstrom a boli si teda geneticky blízke.
- Kvasinky III. Dátová sada kompletných netriedených jadrových sekvencií 10 druhov kvasiniek s dĺžkami 11617kbp až 33625kbp. Druhy, ktoré sekvencie mapujú sú náhodne vyberané z množiny druhov z predošlej dátovej sady. V tomto prípade nemá zmysel uvažovať dĺžky hrán v referenčnom strome, nakoľko tieto boli počítane iba z mitochondriálnych sekvencií. Výsledok z jadrových sekvencií k dispozícií nemáme, a vďaka dĺžke sekvencií, ho ani nie sme schopní získať žiadnym algoritmom so zarovnávaním.

8.2 MIERA PRESNOSTI VÝSLEDKU

Výsledkom algoritmov je dendrogram - *strom života*, ktorý vznikol činnosťou špecifického hierarchického zhlukovacieho algoritmu, do ktorého vstúpila matica rozdielností entít. Túto sme získali špecifickou vzdialenostnou funkciou, resp. kombináciou transformačný filter + vzdialenostná funkcia. Máme teda za cieľ vyhodnotiť jednak presnosť získanej vzdialenostnej matice, ako aj úspešnosť zhlukovacieho algoritmu vo vzájomnej kombinácii.
Z dostupných artefaktov (vzdialenostná matica, výsledný dendrogram) je preto logické na určenie presnosti algoritmu vychádzať z podobností dendrogramov - výsledného a vyššie spomenutého referenčného. Ako

2 Nakoľko algoritmus počítajúci momenty [14] nesprávne normalizuje dĺžku sekvencií, pristúpili sme k ich orezaniu na najmenšiu spoločnú dĺžku.

metriku podobnosti používame dĺžku jednoznačne určenej cesty medzi každou dvojicou dátových entít. Takto získame novú vzdialenostnú maticu, ktorá už ale reflektuje aj párové vzdialenosti entít aj relevantnosť zhlukovania.
Je možné vytvoriť dve verzie - metrickú a diskrétnu. Metrická verzia uvažuje hrany aj s ich dĺžkami, diskrétna uvažuje s dĺžkou každej hrany rovnej jednej a teda reflektuje počet hrán medzi entitami. Matice metrickej verzie je následne nutné normalizovať na (I.) Rovnakú strednú hodnotu aj (II.) Rovnakú smerodajnú odchýlku, aby sa predišlo vplyvu nejednotného mierkovania výsledkov. Normovaním na rovnakú smerodajnú odchýlku sa tiež zabráni falošne pozitívnym výsledkom metód, ktoré nesprávne vyhlasujú všetky entity za rovnako rozdielne, stavu, ktorý je zrejmý z veľmi podobnej hladiny hrán spájajúcich podstromy.

9

VÝSLEDKY A ZÁVER

9.1 TESTOVACIE SCENÁRE

Testovacie scenáre boli navrhnuté pre 9 metód, pričom názvy metód sú skracované nasledovne:

- *A*: Metóda *Frekvencií Slov* s dĺžkou slov v rozsahu $< 4, 8 >$.
- *B*: Metóda *Frekvencií Riedkych Slov* s dĺžkou slov $< 10, 23 >$ s počtom *match* pozícií $< 4, 8 >$.
- *C*: Metóda *Spektrálnej Transformácie podľa Yin et al. [31]*.
- *D*: Metóda *Momentového Spektra podľa Yin et al. [14]*.
- *E*: Metóda *Lokálneho Konvolučného Zarovnávania Sekvencií*.
- *F*: Metóda *Dominantných Spektrálnych Koeficientov* s hranicou dominantnosti v rozsahu $< 5, 20 >\%$[1].
- *G*: Metóda *Spektrálnej Analýzy cez posúvajúce sa okno* s veľkosťou okna v rozsahu $< \min(1024, \frac{l_{seq}}{16}), \min(4096, \frac{l_{seq}}{4}) >$ a $< 0,75 >$ %-ným prekryvom [2].
- *H*: Metóda *MRA Vlnkových Transformácií*
- *I*: Metóda *MRA Spektrálnej Analýzy cez posúvajúce sa okno* s parametrami metódy G a $k = \{4, 8\}$.

9.2 VÝSLEDKY METÓD NA JEDNOTLIVÝCH DÁTOVÝCH MNOŽINÁCH

Metódy sú porovnávané skôr popísanou metrikou, pričom prezentované hodnoty predstavujú percentuálny rozdiel voči presnosti danej náhodnými hodnotami s normálnou distribúciou.

1 F, F1: reprezentácia sekvencií koreňmi jednotky; F2: bipolárna reprezentácia
2 Prekryv do -125% pre jadrové sekvencie

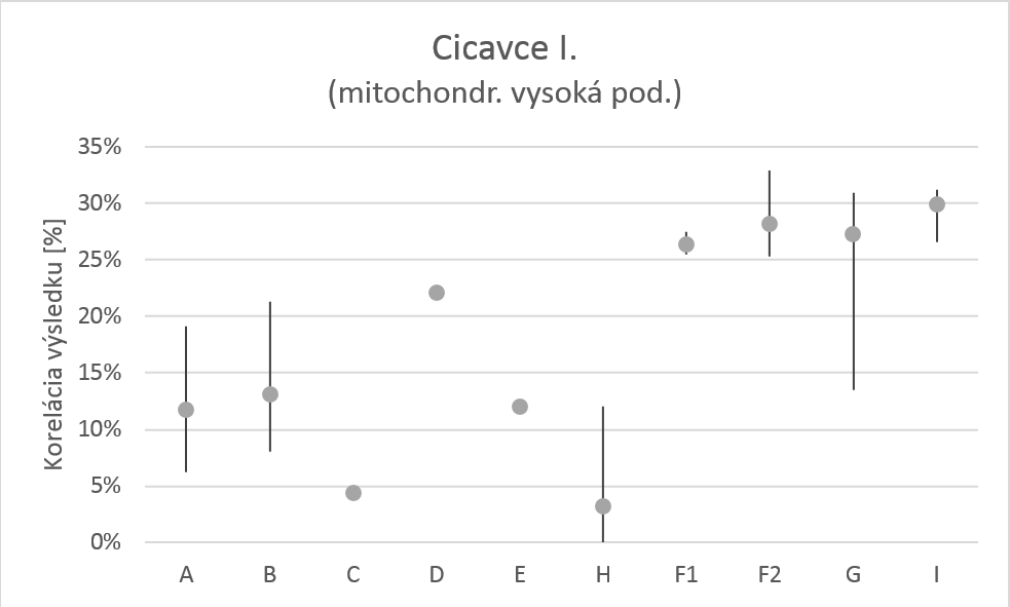
9.2.1 *Dátová množina Cicavce*

Z metód adaptovaných z dostupnej literatúry dosahuje najlepšiu výkonnosť metóda *Momentového Spektra podľa Yin et al.*, z ktorej doplnkových materiálov pochádza dátová sada (22.1%). Nasledujú *Frekvencie Riedkych Slov* (max. 21.2%, min. 8.0%, priem. 13.2%), *Frekvencie Slov* (max. 19.1%, min. 6.2%, priem. 11.8%) a *Spektrálna Transformácia podľa Yin et al.* (4.4%).

Všetky navrhované metódy, ktoré považujeme za perspektívne, dosahujú výrazne lepšie výsledky:

- Metóda dominantných spektrálnych koeficientov - F1 (max. 27.5%, min. 25.5%, priem. 26.4%).
- Metóda F s *bipolárnou reprezentáciou* - F2 (max. 32.9%, min. 25.2%, priem. 28.2%).
- Metóda spektrálnej analýzy cez posúvajúce sa okno - G (max. 19.1%, min. 6.2%, priem. 11.8%).
- MRA metóda - I (max. 19.1%, min. 6.2%, priem. 11.8%).

Na druhej strane, metódy MRA s použitím vlnkových transformácií (H) dosahujú veľmi slabé výsledky, aj so zápornou koreláciou výsledku.

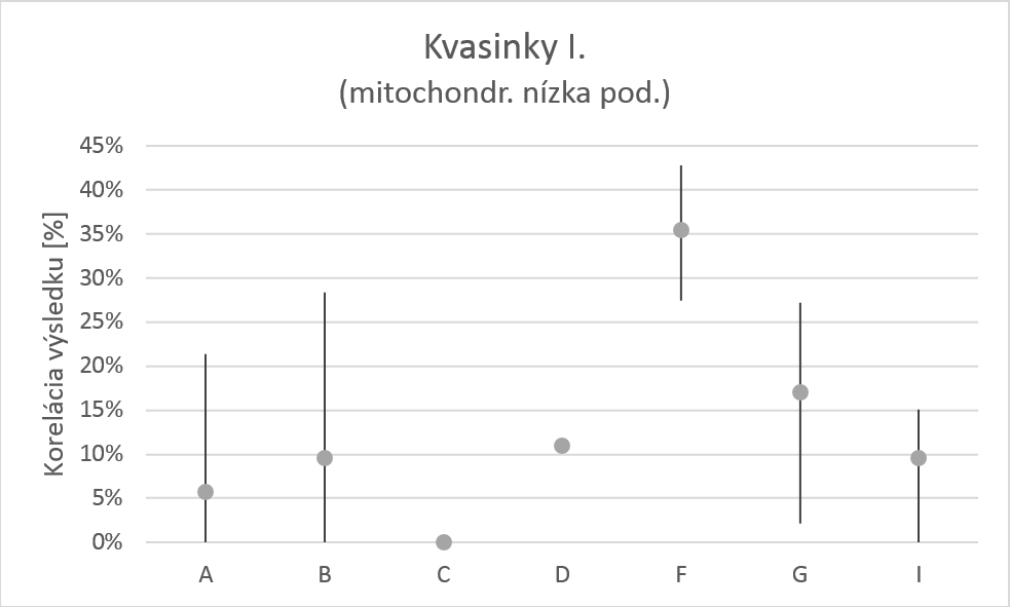


Obr. 20: Presnosť výsledkov na dátovej sade Cicavce I.

9.2.2 *Dátová množina Kvasinky I.*

Dátová sada Kvasinky I. obsahuje sekvencie bez transpozícií s relatívne väčšou vzájomnou rozdielnosťou. Podľa očakávaní, dobré výsledky dosahujú metódy, ktoré spracúvajú celú vzorku a teda dokážu vyhľadať vzory nelimitované veľkosťou okna. Patrí sem metóda D (11.0%), avšak výrazne prekonaná metódou F (max. 42.8%, min. 27.4%, priem. 35.5%). Nadpriemerné výsledky dosahuje aj navrhovaná metóda G (max. 27.2%, min. 2.1%, priem. 17.0%).

Celkovo sa opäť potvrdzuje dominancia navrhovaných metód, najmä metódy F.



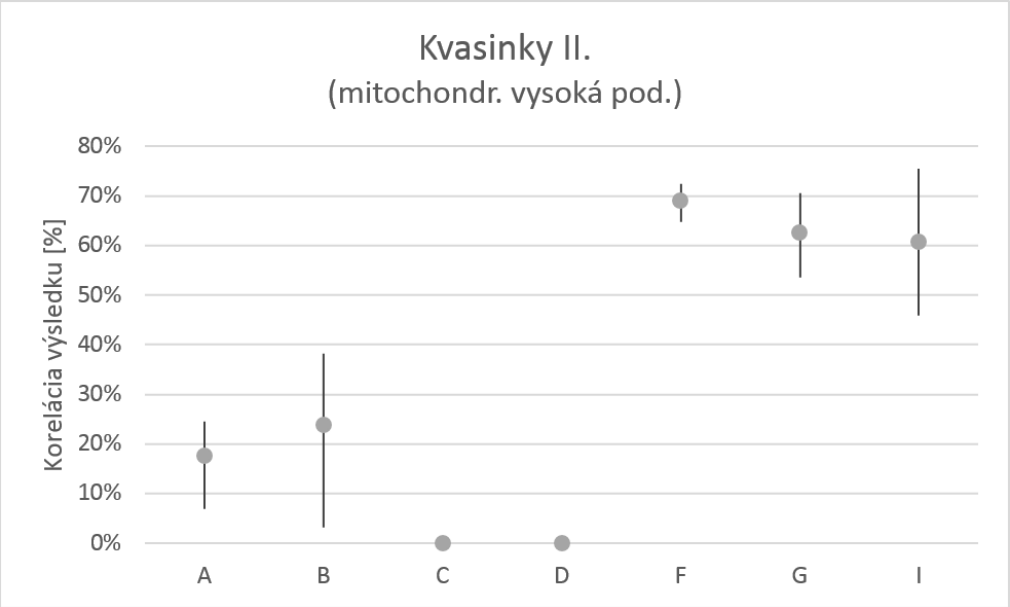
Obr. 21: Presnosť výsledkov na dátovej sade Kvasinky I.

9.2.3 *Dátová množina Kvasinky II.*

Sekvencie v množine Kvasinky II. majú podobné charakteristiky, ako v množine Cicavce. Tomu zodpovedajú aj výsledky, s výnimkou metódy D. Ako bolo spomenuté, táto obsahuje chybu normalizácie dĺžok sekvencií a dokáže správne vyhodnotiť iba dátové sady so sekvenciami rovnakej dĺžky. Bez narušenia presnosti ostatných testov sme toto ale neboli schopní zabezpečiť, preto dosahuje záporné korelácie. Dominuje opäť metóda F (max. 72.4%, min. 64.8%, priem. 69.1%), tesne nasle-

dovaná metódami G (max. 70.6%, min. 53.5%, priem. 62.7%) a I (max. 75.6%, min. 45.6%, priem. 60.9%), ktoré tentoraz netrpia malým rozsahom okna.

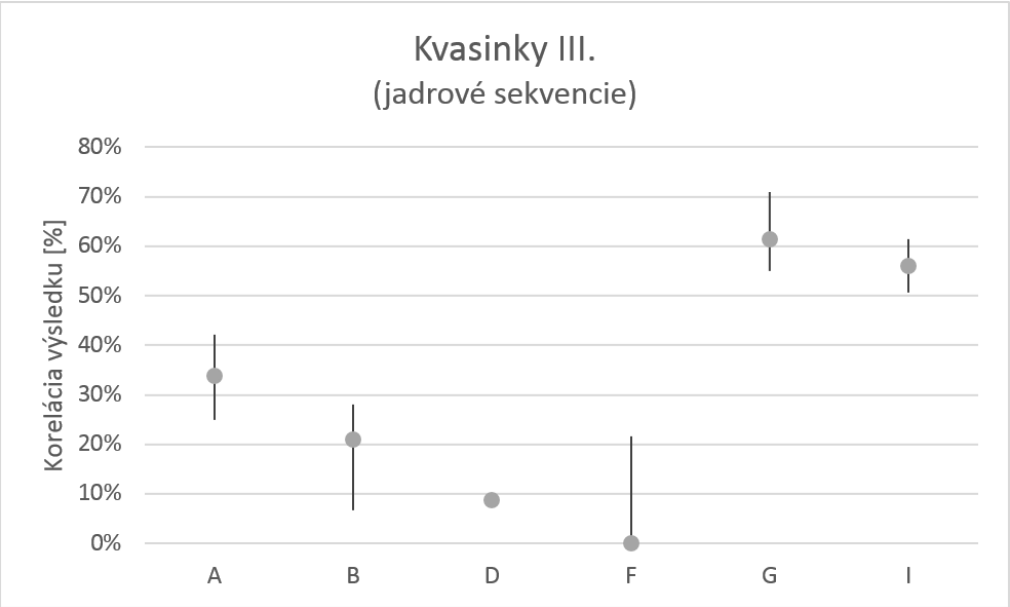
Zaužívané metódy A a B dosahujú oproti nim cca. tretinovú koreláciu výsledku (max. 24.5%, min. 6.8%, priem. 17.7% resp. max. 28.2%, min. 3.2%, priem. 23.8%).



Obr. 22: Presnosť výsledkov na dátovej sade Kvasinky II.

9.2.4 *Dátová množina Kvasinky III.*

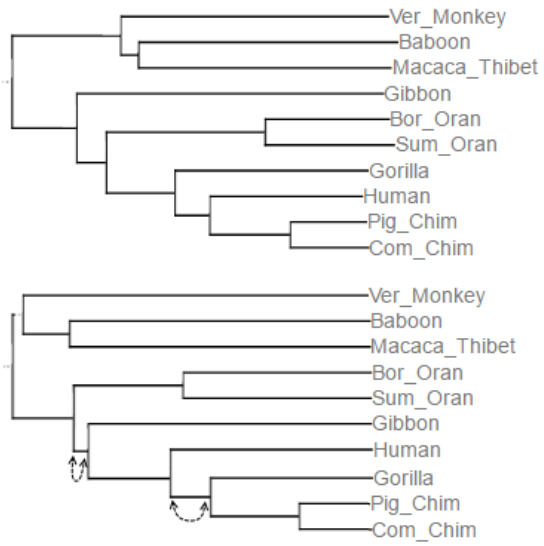
Kvasinky III. sú dátová sada s jadrovými sekvenciami obsahujúcimi transpozície podreťazcov. Očakávame teda značnú nepresnosť metód pracujúcich s celým reťazcom - D a F. V úvahe ostávajú metódy A (max. 42.1%, min. 24.9%, priem. 33.8%) a B (max. 28.1%, min. 6.7%, priem. 21.0%), resp. z navrhovaných metód opäť niekoľkokrát presnejšie G (max. 71.0%, min. 54.9%, priem. 61.4%) a I (max. 64.1%, min. 50.6%, priem. 56.0%).



Obr. 23: Presnosť výsledkov na dátovej sade Kvasinky III.

Testovanie potvrdilo teoretické vlastnosti metód. Na základe nich sme vyhodnotili za najúspešnejšiu metódu *metódu dominantných spektrálnych koeficientov* (F) na krátke mitochondriálne sekvencie bez transpozícií a na jadrové sekvencie *metódu spektrálnej analýzy cez posúvajúce sa okno* (G). Obidve dosahujú, vo svojej kategórii, niekoľkonásobne vyššie hodnoty korelácie, ako zaužívané metódy.

Je nutné ešte podotknúť, že nakoľko do vyhodnotenia presnosti pri mitochondriálnych sekvenciách vstupujú do výsledku aj dĺžky hrán, metódy dosahujú dostatočnú presnosť už na hranici 30% korelácie s referenčným výsledkom.



Obr. 24: Výňatok z výsledku konfigurácie i2 (31.2%) s dvomi chybami - dole, oproti referenčnému grafu - hore.

9.3 VÝSLEDKY TESTOVANIA VPLYVU ČIASTKOVÝCH PARAMETROV

9.3.1 *Časová náročnosť metód*

Výpočtová zložitosť metód s posúvajúcim sa oknom (A, B, G, I) je asymptoticky rovná, v priemere na dátovej množine Kvasinky III. - v poradí 1937, 17693, 15645 a 42835ms Za povšimnutie stojí $\log_2(k)+1)$ - násobný nárast náročnosti metódy I oproti metóde G, v tomto prípade $k = 4$.

Metódy, ktoré pracujú s celým reťazcom sú znevýhodnené náročnosťou spektrálnej transformácie rovnej $O(n\log(n))$ a dosahujú na rovnakých dátach časy spracovania okolo 200s. Nakoľko ale ich jediné zmysluplné použitie je pri pomerne krátkych mitochondriálnych sekvenciách, je táto vlastnosť irelevantná.

9.3.2 *Vplyv typu spektrálnej transformácie*

Pri metóde F signifikantné rozdiely zistené neboli - FFT mierne dominovala nad WHT (max. 6.33%, min. -5.32%, priem. 1.14%). Na metóde

G boli rozdiely v prospech FFT väčšie (max. 12.2%, min. 1.11%, priem. 8.43%).

Vlnkové transformácie boli testované separátne v metóde H a dosiahli kladné korelácie iba s použitím Haarovych ortogonálnych základných funkcií - stále sa však jednalo o podpriemernú hodnotu (12.1%). Ďalšie testované Debouchiovej funkcie dĺžky 20 a biortogálne funkcie dĺžky 35 dosiahli záporné korelácie výsledku.

Všetky ostatné metódy využívali výlučne FFT.

9.3.3 *Vplyv typu numerickej reprezentácie sekvencií*

Testovali sme všetky tri typy reprezentácií, pričom za základ bola vybraná *reprezentácia koreňmi jednotky*. Na testoch metódy F na dátovej množine Cicavce sa prejavila iregularita a dominovala *bipolárna reprezentácia* nad *reprezentáciou koreňmi jednotky* (priem. 3.25%) a *binárnymi indikátormi* (priem. 5.16%).

Pri ostatných dátových sadách a metódach dosahovala *reprezentácia koreňmi jednotky* lepšie výsledky ako *binárne indikátory* (max. 13.4%, min. 4.02%, priem. 10.6%) aj *bipolárna reprezentácia* (max. 41.3%, min. 5.33%, priem. 28.8%).

9.3.4 *Vplyv typu sekundárnej vzdialenostnej funkcie*

Z generických vzdialenostných funkcií sme testovali Kosínusovú uhlovú mieru a Minkowského funkcie: Euklidovskú a Manhattanskú, pričom ako základ bola zvolená Euklidovská vzdialenosť. Táto aj dominovala na všetkých 12 verziách testov oproti obidvom - Manhattanskej (max. 21.5%, min. 11.7%, priem. 17.2%) aj Kosínusovej (max. 0.74%, min. 0.33%, priem. 0.52%). Vo všetkých ostatných testoch sa používa práve Euklidovská vzdialenosť.

9.3.5 *Vplyv typu algoritmu hierarchického zhlukovania*

Do testovania vhodnosti algoritmov hierarchického zhlukovania sme vybrali algoritmy *SLINK* a *Neighbour Joining*. Vo všetkých testovacích

scenároch dominoval algoritmus *SLINK* (max. 49.7%, min. 12.0%, priem. 38.1%). Vo všetkých ostatných testoch sa používa algoritmus *SLINK*.

9.3.6 *Vplyv použitia zahladzovacieho okna*

Zahladzujúce okno má význam pri metódach, ktorých súčasťou je doplnenie sekvencií nulami na požadovanú dĺžku. Testy sme teda vykonali na metóde *Dominantných Spektrálnych Koeficientov*. Prvé tri konfigurácie porovnávajú výsledné vektory Euklidovskou vzdialenostnou funkciou a po úprave dosahujú v priemere horšie výsledky (max. 19.4%, min. -0.5%,priem. 10.8%).
Druhé tri konfigurácie používajú vzdialenostnú funkciu sledujúcu počet zhôd na pozíciách dominantných spektrálnych koeficientov (podľa popisu v časti 5.3.2). Podľa očakávaní zahladenie v spektrálnej oblasti (viď časť 3.3), prinieslo výrazné zlepšenie (max. 18.9%, min. 9.7%, priem. 15.3%).

9.4 ZÁVER

Na základe experimentov môžeme potvrdiť, že je možné efektívne vyhodnocovať vzájomnú podobnosť sekvencií DNA s využitím princípov signálových transformácií. Navrhované metódy dosahujú signifikantne väčšiu presnosť výsledku, ako niekoľko iných metód, a to najmä z dôvodu, že vo väčšej miere zohľadňujú špecifiká DNA sekvencií. Pritom však nekladú výrazne vyššie nároky na výpočtové kapacity.

Zároveň sa nám podarilo zanalyzovať rozšíriteľnú platformu pre dolovanie v dátach. Táto analýza odhalila výhody existujúceho riešenia, ale aj možné zlepšenia a môže tak poslúžiť pri plánovanom vývoji distribuovaného softvérového riešenia na dolovanie v dátach.

LITERATÚRA

[1] Allhoff, M., Schonhuth, A., et al. Discovering motifs that induce sequencing errors. *BMC Bioinformatics 14*, Suppl 5 (2013), S1+.

[2] Almeida, J. S. Sequence analysis by iterated maps, a review. *Briefings in Bioinformatics 15*, 3 (2014), 369.

[3] Bey, N. Y. Multi-resolution fourier analysis: achieved high resolutions with suppressed finite observation effects. *Signal, Image and Video Processing 10*, 4 (2016), 711–718.

[4] Cooley, J. W., and Tukey, J. W. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation 19*, 90 (1965), 297–301.

[5] Edgar, R. C. Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics 5*, 1 (2004), 113.

[6] Fahad, A., Alshatri, N., Tari, Z., et al. A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. *IEEE Transactions on Emerging Topics in Computing 2*, 3 (Sept 2014), 267–279.

[7] Goel, S., Kaur, G., and Tomar, P. Performance analysis of welch and blackman nuttall window for noise reduction of ecg. In *2015 International Conference on Signal Processing, Computing and Control (ISPCC)* (Sept 2015), pp. 87–91.

[8] Gronau, I., and Moran, S. Optimal implementations of upgma and other common clustering algorithms. *Inf. Process. Lett. 104*, 6 (Dec. 2007), 205–210.

[9] Han, J., and Kamber, M. *Data Mining: Concepts and Techniques*, 3 ed. Morgan Kaufmann, Jan. 2012.

[10] Hanson, R. Fast Fourier Transform Analysis of DNA Sequences, Thesis, Reed College (Oregon, USA), 2003.

[11] Harris, F. J. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE 66*, 1 (Jan. 1978), 51–83.

[12] Haubold, B. Alignment-free phylogenetics and population genetics. *Briefings in Bioinformatics 15*, 3 (2014), 407.

[13] Hazewinkel, M. *Encyclopaedia of Mathematics*. Encyclopaedia of Mathematics: An Updated and Annotated Translation of the Soviet "Mathematical Encyclopaedia". Springer, 2001.

[14] Hoang, T., Yin, C., Zheng, H., et al. A new method to cluster DNA sequences using Fourier power spectrum. *Journal of Theoretical Biology 372* (2015), 135 – 145.

[15] Kantorovitz, M. R., Robinson, G. E., and Sinha, S. A statistical method for alignment-free comparison of regulatory sequences. In *Proceedings 15th International Conference on Intelligent Systems for Molecular Biology (ISMB) & 6th European Conference on Computational Biology (ECCB), Vienna, Austria, July 21-25, 2007* (2007), pp. 249–255.

[16] Karp, R. M., and Rabin, M. O. Efficient randomized pattern-matching algorithms, 1987.

[17] Katoh, K., and Toh, H. Recent developments in the MAFFT multiple sequence alignment program. *Briefings in Bioinformatics 9*, 4 (July 2008), 286–298.

[18] Leimeister, C., Boden, M., Horwege, S., et al. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics 30*, 14 (2014), 1991–1999.

[19] Leimeister, C.-A., and Morgenstern, B. KMACS: The k-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics 30*, 14 (July 2014), 2000–2008.

[20] Ouyang, W. *Fast Pattern Matching and Its Applications*. PhD thesis, The Chinese University of Hong Kong (People Republic of China), 2011. AAI3492004.

[21] Ouyang, W., and Cham, W. Fast algorithm for Walsh Hadamard transform on sliding windows. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 1 (Jan. 2010), 165–171.

[22] Rozinaj, G. Vlastnosti a využitie matematických transformácií v procese spracovania signálov: Osobná konzultácia, 18.4.2016.

[23] Schwende, I., and Pham, T. D. Pattern recognition and probabilistic measures in alignment-free sequence analysis. *Briefings in Bioinformatics 15*, 3 (2014), 354.

[24] Sibson, R. Slink: An optimally efficient algorithm for the single-link cluster method. *Comput. J. 16*, 1 (1973), 30–34.

[25] Song, K., Ren, J., Reinert, G., et al. New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Briefings in Bioinformatics 15*, 3 (2013), 343.

[26] Valach, M., Farkas, Z., Fricova, D., et al. Evolution of linear chromosomes and multipartite genomes in yeast mitochondria. *Nucleic Acids Research 39*, 10 (2011), 4202–4219.

[27] Vinga, S. Information theory applications for biological sequence analysis. *Briefings in Bioinformatics 15*, 3 (2014), 376.

[28] Vinga, S., and Almeida, J. Alignment-free sequence comparison-a review. *Bioinformatics (Oxford, England) 19*, 4 (Mar. 2003), 513–523.

[29] Witten, I., Frank, E., and Hall, M. *Data Mining: Practical Machine Learning Tools and Techniques*, 3 ed. Morgan Kaufmann, Jan. 2012.

[30] Yin, C. Identification of repeats in DNA sequences using nucleotide distribution uniformity. *CoRR abs/1608.00567* (2016).

[31] Yin, C., Chen, Y., and Yau, S. S.-T. S. A measure of DNA sequence similarity by Fourier Transform with applications on hierarchical clustering. *Journal of theoretical biology 359* (Oct. 2014), 18–28.

[32] Yin, C., and Wang, J. A novel method for comparative analysis of DNA sequences by ramanujan-fourier transform. *CoRR abs/1403.1523* (2014).

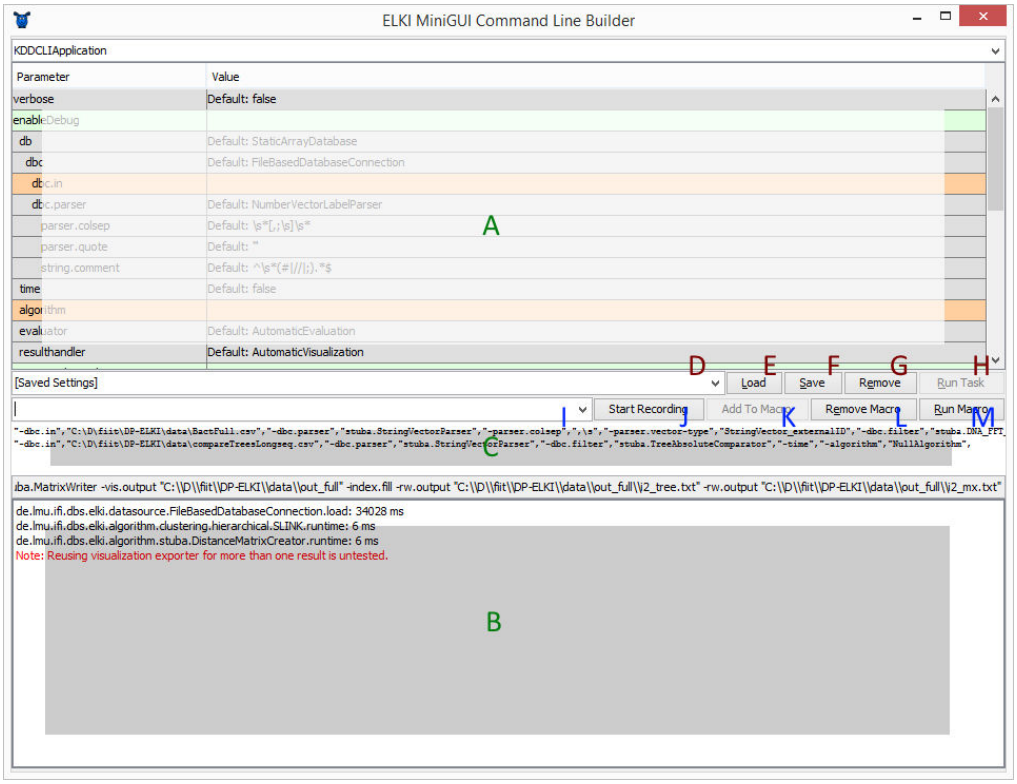
[33] Zendulka, J., et al. *Získávání znalostí z databází. ZZN. Studijní opora. FIT VUT v Brně.* FIT VUT v Brně, 2010.

A

PRÍLOHA A: POUŽÍVATEĽSKÁ PRÍRUČKA

ELKI MINIGUI COMMAND LINE BUILDER

Platforma ELKI je distribuovaná s GUI rozhraním uľahčujúcim tvorbu príkazov, ktorá sa majú spustiť.



Obr. 25: Časti okna aplikácie ELKI.

Skladá sa z nasledovných súčastí:

A: Panel parametrov konfigurácií (editovateľný).

Nakonfigurovanú úlohu je možné uložiť pre neskoršie spustenie. Na tento účel slúži panel manažmentu úlohy.

D: Selektor názvu úlohy (výber existujúceho alebo vpísanie nového názvu).

E: Tlačidlo načítania vybranej konfigurácie do panela A.

F: Tlačidlo uloženia aktuálnej konfigurácie z panela A do vybraného názvu.

G: Tlačidlo odstránenia úlohy pod vybraným názvom.

H: Tlačidlo spustenia aktuálnej úlohy (podľa obsahu panela A).

Na zjednodušenie spúšťania dávkových súborov úloh je možné zoskupiť viacero úloh do jednej dávky pomocou Panel manažmentu dávkového súboru úloh (makra).

C: Výstup aktuálnej konfigurácie (editovateľný, no bez validácie).

I: Selektor názvu dávkovej úlohy, analogický s komponentom D.

J: Tlačidlo vytvorenia makra.

K: Tlačidlo na pridanie aktuálnej úlohy do makra.

L: Tlačidlo odstránenia makra pod vybraným názvom.

M: Tlačidlo spustenia aktuálneho makra (podľa obsahu panela C).

Textový výstup algoritmov (systémový výstup, chybové výstupy) sa presmerováva do okna grafického rozhrania:

B: Panel textového výstupu algoritmov.

VYTVORENIE A SPUSTENIE ÚLOHY

1.
   - Konfigurácia úlohy sa vykonáva nastavením parametrov v paneli "A". Povinné parametre sú zvýraznené oranžovou farbou, počet a štruktúra parametrov sa dynamicky mení.
   - Obsah panelu "A" je možné načítať z uložených konfigurácií. V tomto prípade používateľ vyberie v selektore "D" konfiguráciu podľa názvu a tlačidlom "Load - E" nastaví parametre v paneli A podľa nej.
2. Po vyplnení panelu konfigurácie "A" sa odporúča aktuálnu konfiguráciu uložiť pre potreby neskoršieho spustenia. Používateľ vyplní názov konfigurácie do poľa "D" a konfiguráciu uloží stlačením tlačidla "Save - F". Pokiaľ sa názov konfigurácie zhoduje s niektorou zo skôr uložených konfigurácií, táto sa prepíše aktuálnou.

3. Tlačilo "Run Task - H" spustí úlohu danú konfiguráciou nastavenou v paneli konfigurácie "A". Validácia parametrov prebieha pri ich nastavovaní, pričom spustenie úlohy je možné iba v prípade, že konfigurácia je správna a úplná. V inom prípade je tlačidlo "H" uzamknuté. Pokiaľ používateľ menil niektoré parametre medzi načítaním a spustením úlohy, musí pamätať na opätovné uloženie konfigurácie, keďže toto sa pri spustení automaticky nedeje.

Konfiguráciu podľa názvu v selektore "D" je možné odstrániť tlačidlom "Remove - G".

VYTVORENIE A SPUSTENIE DÁVKOVEJ ÚLOHY

Dávková úloha sa vytvára obdobne ako bežná úloha kompozíciou bežných úloh do jedného súboru.

1. Tlačidlom "Start Recording - J" sa aktivuje ukladanie novej dávkovej úlohy - makra.
2. Tlačidlo "Add To Macro - K" pridá aktuálne konfigurovanú úlohu podľa panelu konfigurácie "A" do aktuálneho makra. Akcia sa prejaví pridaním konfigurácie do panelu makra "C"
3. Tlačidlo "Save Macro - J" uloží aktuálnu konfiguráciu makra podľa názvu v poli "I", existujúce makro sa prepíše.
4. Tlačidlo "Run Macro - M" spustí makro.

Makro podľa názvu v selektore "I" je možné odstrániť tlačidlom "Remove Macro - L".

POUŽITIE V KONTEXTE BIOINFORMATICKÝCH APLIKÁCIÍ

Vytvorenie úlohy s transformačným filtrom na jednoduchých sekvenciách

Spustenie úlohy s transformačným filtrom na jednoduchých sekvenciách predstavuje scenár, pri ktorom je na vstupe n entít, z ktorých každá je popísaná práve jednou sekvenciou. Úloha zahrňuje načítanie metadát o entitách, vytvorenie transformovaných numerických vektorov z entít (vrátane načítania sekvencií podľa metadát) a kompletné

hierarchické zhlukovanie a výstup dát. Nastavenie parametrov je nasledovné:

Načítanie metadát

- *db*: *StaticArrayDatabase*
- *dbc*: *FileBasedDatabaseConnection*
- *dbc.in*: cesta k súboru s metadátami, viď vzor
- *dbc.parser*: *stuba.StringVectorParser*
- *dbc.vector-type*: *StringVector_externalID*

Vykonanie transformačnej filtrácie

- *dbc.filter*: Vybraná trieda filtra [1]
- *filter.dataProvider*: *FastaDnaDataProvider*
- parametre špecifické zvolenému filtru: Podľa popisu filtra

Vykonanie hierarchického zhlukovania

- *algorithm*: *clustering.hierarchical.SLINK* alebo ľubovoľný iný. [2]
- *algorithm.distancefunction*: *minkowski.EuclideanDistanceFunction*[3] alebo ľubovoľná iná.

Výstup dát

- *resulthandler*: *stuba.ExportLabelledVisualizations* [4] [5]

1 Popisy entít sú dôležité pre správny výpis alebo vykreslenie grafov. V prípade, že algoritmus filtra neobsahuje časť extrahujúcu popisy entít (situácia sa prejaví chybou *No data for type LabelList found*), je nutné pridať filter *StringVectorToLabelFilter*.

2 V prípade, že používateľ chce výstup aj vo forme vzdialenostnej matice, je nutné doplniť *stuba.DistanceMatrixCreator*. Vzdialenostné matice ako výstup viacerých filtrov je možné neskôr kombinovať v inej úlohe.

3 Výstup *stuba.ST_Filter* vyžaduje porovnávaciu funkciu *stuba.SparseVectorComparator*

4 Pokiaľ používateľ chce aj výstup dendrogramu vo forme matice traverzových vzdialeností, je nutné doplniť *stuba.BinaryTreeDistanceWriter*

5 Pre výstup matice vytvorenej *stuba.DistanceMatrixCreator* je nutné doplniť *stuba.MatrixWriter*.

- *rw.output*: Cesta k výstupnému adresáru. [6]

Vytvorenie úlohy s transformačným filtrom na entitách s viacerými sekvenciami

Transformácia entít, z ktorých každá je popísaná k sekvenciami je realizovaná po sekvenciách, pričom výsledkom transformácie je artefakt obsahujúci k výsledkov z každej entity. Uvádzame parametre rozdielne oproti minulej úlohe:

Načítanie metadát

- *dbc.vector-type*: *StringVector_externalMultipleID*

Vykonanie hierarchického zhlukovania

- *algorithm.distancefunction*: *stuba.MultipleTypeVectorComparator* [7]
- *dist.functions* - zoznam k vzdialenostných funkcií; pre každý vektor v artefakte.
- *dist.weights*: Zoznam váh, aké prislúchajú výsledkom jednotlivých vzdialenostný funkcií. Východzie nastavene je jednotkový vektor.

Vytvorenie úlohy s externou vzdialenostnou funkciou

Za účelom zoskupenia viacerých vzdialenostných matíc (ako výsledku rôznych vzdialenostných funkcií a filtrov s rôznymi parametrami) je možné načítať viacero skôr vytvorených matíc a agregovať ich. Konfigurácia sa líši v nasledovnom:

Vykonanie transformačnej filtrácie

- *dbc.filter*: *StringVectorToLabelFilter*; na vytvorenie zoznamu popisov entít, keďže žiaden iný filter sa nepoužíva.

6 ELKI obsahuje vadu, ktorá sa prejavuje nutnosťou vybrať súbor, nie adresár. Zvolenú cestu pre *ExportVisualizations* alebo *stuba.ExportLabelledVisualizations* je nutné vybrať ako súbor a následne textovo doeditovať na obsahujúci adresár.

7 Algoritmus pracuje aj s artefaktom obsahujúcim k vektorov rôznych typov

Vykonanie hierarchického zhlukovania

- *algorithm.distancefunction*: *extrnal.stuba.AggregativeFileBasedMatrixDistanceFunction*
- *distance.matrix*: Zoznam ciest k súborom s maticami vytvorenými DistanceMatrixCreator
- *distance.mxWeights*: Váhy jednotlivých matíc do celkového výsledku.

B

PRÍLOHA B: DOKUMENTÁCIA K SPUSTENIU APLIKÁCIE A OBSAH PRILOŽENÉHO NOSIČA

DOKUMENTÁCIA K SPUSTENIU APLIKÁCIE

Programovací jazyk a platforma

Aplikácia je implementovaná v jazyku Java. Beh aplikácie s využitím doimplementovaných súčastí vyžaduje JRE min. verzie 1.8. Jadro aplikácie požaduje JRE min. verzie 1.7.
Hlavná spustiteľná trieda (grafické používateľské rozhranie) je *elki-guiminigui/de.lmu.ifi.dbs.elki.gui.minigui.MiniGUI*. Spustenie nevyžaduje žiadne parametre v príkazovom riadku. Konfigurácie 3d∗ a 3f∗ vyžadujú min. 3GB pamäte RAM, aplikáciu je preto potrebné spustiť s prepínačom *-Xmx3g*

- *data*: Adresár obsahuje vstupné dáta do aplikácie. Súbory s príponou **.csv* špecifikujú vstupné entity. Súbory s príponou **.fasta* obsahujú sekvencie DNA referencované v dátových entitách.
  - *out_x*: Výstupné súbory (matice podobností, dendrogramy, vyhodnocovacie tabuľky) pre dátové sady.
- *source*: Adresár obsahuje zdrojové súbory aplikácie
  - *changed_files.txt*: Zoznam súborov upravených voči východzej verzii ELKI.
  - *dp-elki*: Repozitár aplikácie ELKI.
  - *jWawe*: Knižnica pre optimalizované počítanie transformácií.
- *doc*: Dokumentácia k dielu
  - *5220_52026.pdf*: Elektronická verzia tohto dokumentu.
  - *paper_IITSRC.pdf*: Článok na študentskú vedeckú konferenciu IIT.SRC 2017.
  - *paper_full.pdf*: Plná verzia pripravovaného článku do vedeckého žurnálu.
  - *poster.pdf*: Poster k článkom.

C

PRÍLOHA C: VYHODNOTENIE PLÁNU PRÁC

Úloha	Plnenie	Pribl. dátum
Analytické úlohy		
Špecifikácia problému na riešenie	Splnená	11/2015
Analýza zhlukovacích algoritmov	Splnená	10/2015
Analýza prístupov porovnávania DNA	Splnená	02/2016-03/2016
Analýza signálových transformácií a ich vlastností	Splnená	11/2015-11/2016
Návrhové úlohy		
Návrh vlastných algoritmov: Transformačné funkcie	Splnená	03/2016-11/2016
Návrh vlastných algoritmov: Transformačné filtre	Splnená	04/2016-12/2016
Implementačné úlohy		
Oboznámenie sa s platformami analýzy dát (ELKI, Weka)	Splnená	12/2015
Hlbšie štúdium vybranej platformy (ELKI)	Splnená	03/2016-05/2016
Analýza ďalších požiadaviek na ELKI	Splnená	04/2016
Implementácia ďalších požiadaviek na ELKI	Splnená	04/2016-07/2016
Implementácia transformačných funkcií	Splnená	09/2016-10/2016
Implementácia transformačných filtrov	Splnená	10/2016-12/2016
Dokumentačné úlohy		
Príprava testovacích dát	Splnená	08/2016-12/2016
Prvá iterácia testovania	Splnená	09/2016-12/2016
Optimalizácia parametrov komponentov	Splnená	01/2017
Finálna iterácia testovania	Splnená	01/2017-03/2017
Príprava príspevku IIT.SRC	Splnená	02/2017
Finalizácia výsledného dokumentu	Splnená	04/2017
Príprava príspevku do vedeckého časopisu	Prebieha	07/2017

D

PRÍLOHA D: OSTATNÉ TLAČENÉ VÝSTUPY

- Článok do bioinformatického časopisu *Bioinformatics (Oxford Academic Journals)* - predbežná verzia v úplnom znení.
- Poster prezentovaný na študentskej vedeckej konferencii IIT.SRC 28.4.2017.

Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

Data and text mining

Novel DNA alignment-free comparison methods based on signal processing approaches

Tomas Farkas [1,*], Maria Lucka [1]

[1]Faculty of Informatics and Information Technologies, Slovak University of Technology, Bratislava, 84216, Slovakia

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Computing similarity between two nucleotide sequences is one of fundamental problems in bioinformatics. Current methods are based on two major approaches: Full sequence alignment, which is computationally expensive, and faster, but less accurate alignment-free methods based on various statistical summaries, e.g. short word counts.
Results: We propose three novel methods based on signal processing transforms designed for overcoming specific sequence features while requiring only modest computational resources. Our approaches include spectral transforms computed across a smoothed sequence, sliding windows or multiple resolution windows. The experiments reveal that the novel methods are up to three times more accurate than current alignment-free methods, while they are equally computationally inexpensive.
Availability: Source files and sample data are available at https://bitbucket.org/bioInfoFiitStu/elki-dna
Contact: xfarkast@stuba.sk

1 Introduction

Computing similarity or distance between two nucleotide sequences is one of the basic steps in many areas of bioinformatics. One application is construction of phylogenetic trees which represent evolution of a group of species in a dendogram (tree diagram). For large datasets such trees are often constructed by distance-based methods, which apply variants of hierarchical clustering to a distance matrix expressing dissimilarity between each pair of species. Distance-based methods include Neighbor Joining (Saitou and Nei, 1987), UPGMA (Gronau and Moran, 2007) or SLINK (Sibson, 1973). The distance matrix is typically computed by comparing nucleotide or protein sequences representing individual species. So there is a need to develop an efficient method that can evaluate pairwise similarity or distance between pairs of sequences to provide data for dendogram construction (Haubold, 2014). However differences between individual sequences acquired during evolution can have many forms, including substitutions of individual nucleotides or amino acids, insertions or deletions of shorter sequences, as well as duplications and rearrangements of sequence parts (Haubold, 2014; Borrayo *et al.*, 2014). These rearrangements can cause a sequence region to be located at a different position compared to its similar counterpart in another sequence. As a result, spatial information[1], which is the only one we have, needs to be processed in a way that tolerates such changes.

[1] In signal processing terminology - positions of characters in the signal sequence.

2 Related Work

Sequence comparison methods in bioinformatics can be divided to two major groups (Haubold, 2014): (I.) Alignment methods, that create local pairwise alignment of sequences and evaluate the number of substitutions between two entities (Haubold, 2014). (II.) Alignment-free methods (Schwende and Pham, 2014), which are heuristics based mainly on statistical characteristics of data (Vinga and Almeida, 2003). Established methods in this group include for instance trivial *Word frequencies method* (Kantorovitz *et al.*, 2007) (A_WFM), that counts the number of occurences for all possible sequences of fixed length ($l =< 4, 8 >$) across the genome and then compares the resulting frequency vectors. Based on A_WFM, *Spaced Word Frequencies Method* (Leimeister *et al.*, 2014) (B_SWFM) has been developed later, utilizing a different frequency vector computation method. In this case, the *word* is not a short contiguous sequence, but a sequence of characters with fixed preset distances in the original genome. There is also a similar method that counts *di-, tri-, tetra-nucleotide abundance* using a more sophisticated formulae (Song *et al.*, 2013).

Another well-known method searches for average longest common substring or average common substring with fixed k mismatches among pairwise considered sequences. The computation needs to be performed strictly pairwise, leading to higher computational cost, although its performance is comparable (Leimeister and Morgenstern, 2014) to B_SWFM.

Group of novel non-alignment methods proposed recently (Yin *et al.*, 2014; Hoang *et al.*, 2015) use spectral transforms instead of simple statistical quantities. Application of spectral transforms, mainly Fast Fourier Transform (Cooley and Tukey, 1965) (FFT) to DNA sequences is

not a new technique. Well established sequence alignment method *MAFFT* (Katoh and Toh, 2008) uses Fourier transform and convolution principle to find local matches. The process is, however, still strictly pairwise and therefore slow. Since 2014, Yin et al. (Yin *et al.*, 2014) have been using spectral transforms to relax strict dependency on spatial information contained in sequences. Their methods compare raw signal spectra acquired from original sequences, or compute numerical characteristics out of raw signal spectra, referred as *Signal Moments method* (Hoang *et al.*, 2015) (C_YMM). The method is claimed to be very accurate, however it includes a severe fault, as the spectral coefficients are not normalized by the input sequence length. According to Parseval's theorem, the computed coefficients depend on the sum of input sequence indicators and so without proper normalization, its results reflect predominantly the input sequence length, not the computed coefficients.

3 Background

3.1 Spectral transforms

Spectral transform is a transformation of an input numerical series of length N which represents time or spatial domain into signal spectrum - N new numerical values in frequency domain. Transformation itself discovers exact or approximate repeats of any size and evaluates their accuracy level. In the frequency domain, the repeat distance is represented by an index in the output vector and the cumulative accuracy level by the corresponding number located in the output vector at that index. As a result, transform can detect and represent all kinds of repetitions including short *codons* (Yin *et al.*, 2014) up to differences in statistic nucleotide distribution among sequence parts or even chromosomes.

Discrete Fourier Transform is one of the well known spectral transforms defined as follows:

$$F(u) = \sum_{x=0}^{N-1} f(x) e^{-i2\pi ux/N}$$

Where $F(u) \in < 0, N-1 >$ is the u-th element of the signal spectrum and N is the length of the input vector f. By definition, the transform approximates input signal using series of sine and cosine functions with period $< 0, N-1 >$, or, using another point of view, computes a convolution of series of sine and cosine waves with different periods and the input vector creating so a series of numerical values. Transformation can be efficiently computed using $O(n \log(n))$ Fast Fourier Transform algorithm (**?**).

Walsh-Hadamard transform (WHT) (Ouyang and Cham, 2010) approximates signal by rectangular basic functions and therefore can be computed using less computationally expensive addition and subtraction operations. A version optimized for sliding window processing also exists (Ouyang and Cham, 2010). The transform was investigated here as a possible replacement for FFT.

3.2 Spectral transforms in discovering motifs in sequences

A DNA sequence can be expressed as a non-periodic signal with some periodic repetitive parts (Yin, 2016). Transforming a non-periodic signal into a signal spectrum may resemble hashing one representation to another without understanding the internal structure and motifs. The needed background can be explained as finding variations in the density of characters inside a sequence. Every spectral coefficient will reflect differences considering a period related to its index ($\lambda = \frac{N}{u}$). Zero coefficient ($u = 0$) reflects the overall sum of the input vector elements, lower coefficients (low u) reflect differences separated by long distance and higher coefficients analyze short sequence parts.

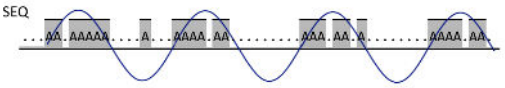


Fig. 1. Visualization of single spectral coefficient calculation from a signal sequence. In this case, the spectral coefficient for the period of 16 characters ($F(\frac{N}{16})$) will have significant non-zero value as positive base function parts are convolved with non-zero signal parts.

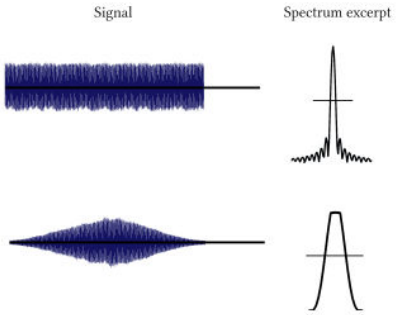


Fig. 2. Visualization of spectral peaks computed from zero-padded signal with and without smoothing window. Smoothing the input vector typically leads to a peak with reduced side-lobes, wider spread and some scalloping loss (truncation of the top).

Typical spectral transform is used for periodic signals and therefore is not localized in time (position). As a result, spectral coefficients are computed using a series of successive base function periods, see Figure 1 showing three such periods. Short motifs are however typically separated by random distances, not necessarily the distance reflecting some equal period lengths. These random distribution can cause canceling interference and harm the results. The situation is similar if two sequences differ in a short insertion or deletion that shifts a whole sequence part. Avoiding these problems is the main motivation for Multiple Resolution Analysis (MRA) approaches as discussed later.

3.3 Padding and Smoothing window

Abrupt truncation of input signal may lead to creating false side lobes neighboring the peaks in the spectral domain. However, input vectors need to be padded with zero values because of: (I.) FFT's ability to handle only inputs of 2^p in length. (II.) Need to scale the resulting spectra to an equal length or resolution.

To overcome this problem, the input signal is convolved with non-rectangular window, which is the process referred as *windowing*. By convolution theorem, windowing in spatial domain results in smoothing in frequency domain. So side lobes are reduced into one smoothened peak at the correct position.

Various types of smoothing windows (Harris, 1978) cause different smoothing behavior. For our purposes, we are interested in the following characteristics:

- Side-lobe reduction to eliminate false lobes at least under a certain threshold.
- Maximal peak width.
- Any amount of scalloping loss.

The purpose will be clear later, neverlethess one of the windows that meet our requirements is Blackman-Nutall window (BNW) (Goel *et al.*, 2015).

4 Method Design

In this section we propose and discuss performance and suitability of three new methods designed to overcome DNA data specifics. Each of the methods is proposed for a group of data with some common quality:

- *Dominant Spectral Coefficient Filter Method* (D_DCM), which is very accurate on input data with none or few rearrangements.
- *Sliding Window Spectral Filter Method* (E_SWM) sequences with frequent rearranged parts.
- *MRA Sliding Window Spectral Filter Method* (F_MSWM) for nuclear sequences with both rearrangements and insertions or deletions.

4.1 Overall architecture

High computational complexity of alignment methods is caused by high complexity of alignment itself, but also by strict pairwise approach. The dissimilarity matrix for n entities contains n^2 elements. Its computation needs to consider each pair of entities, implying $O(n^2)$ time complexity. Although not obvious, some methods are able to produce $O(n^2)$ pairwise distances with dominating $O(n)$ linear time. These include all methods that perform time-consuming sequence analysis once for each entity producing so vector of characteristics with reduced dimensionality. These include for instance *Word frequencies* (Vinga and Almeida, 2003) producing vector of quantities with typical length of $< 4^4, 4^8 >$ or *Yin Signal Moments method* (Hoang *et al.*, 2015) producing just twelve representative numbers for each input sequence. Pairwise comparison is then performed using generic distance functions (Vinga and Almeida, 2003) on a much smaller data and therefore the $O(n^2)$ part does not dominate. All our new methods use this pattern, hence their names include *projection* to express the nature of projecting input sequences into numerical vectors.

4.2 Dominant spectral coefficient projection method

Dominant spectral coefficient method handles input sequences in the whole, so like Yin's methods (Yin *et al.*, 2014; Hoang *et al.*, 2015; Yin and Wang, 2014) the method is not suitable for sequences with rearranged parts. Data is processed as follows:

1. Input DNA sequences are transformed into theirs numerical representations. Yin et al. use *Binary Sequence Indicators* (BSI) creating c sequences, where c is the input sequence alphabet cardinality. Each sequence contains a positive integer on the positions where the selected character matches the original sequence and 0 otherwise. The c vectors are then processed separately implying at least c-times higher computational complexity.

   Our methods represent sequences by complex root-of-one coefficients, considering basis complements. Hence the mapping is as follows: $A = 1, T = -1, C = i, G = -i, N = 0$. The representation has some advantages: It maps a sequence into a single vector and is able to represent unknown input character *N* without affecting results. The BSI representation was considered and tested as well, however giving less accurate results[2].

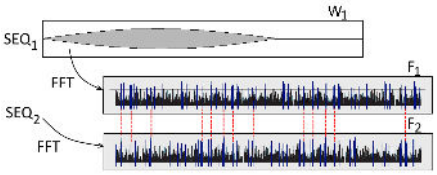


Fig. 3. Dominant spectral coefficient projection process visualization.

2. The single resulting vector is smoothed using BNW and padded with zeros to a maximal common sequence length given as a parameter.
3. The numerical representative vector is transformed into signal spectra using the desired spectral transform. Method is designed to be able to use both Fast Fourier and Walsh-Hadamard Transforms, while FFT seems to produce more accurate results[3].
4. The resulting spectrum is searched for dominant peaks - spectral coefficients with the highest absolute values. The fraction of coefficients to be selected as dominant is given as a parameter. Process creates a vector of positions where the dominant coefficients occur. As all input sequences have been padded to an equal length, the resulting spectra are also of the same length, and therefore no normalization is needed.
5. Pairwise comparison is performed by evaluating the number of matches on dominant positions between entities. Considering both positions and coefficient values themselves was also tested, however with less accurate results.

Blackmann-Nutall smoothing window mentioned previously plays a vital role in this method. Using windowing, peaks in the power spectrum expand into neighboring indices providing a slight mismatch tolerance. However, as the signal spectrum is computed across the whole sequence, sequences with transposed parts are not processed correctly. Because of $O(n \log(n))$ nature of FFT compared to $O(n)$ complexity of windowed methods (later), the method has higher complexity on long sequences with more than $10Mbp$ in length.

4.3 Sliding window spectral projection method

Sliding window spectral projection method (E_SWM) is a transformation of a nucleotide sequence to a representative numerical vector of reduced dimensionality produced by the following steps:

1. Sequence represented by root-of-one numerical vector is splitted into windows of arbitrary predefined sizes and overlaps.
2. Each window is processed separately by selected spectral transform. The set of transforms again contained FFT and less accurate WHT.
3. All vectors containing spectral analysis, one per window, are summed to a single resulting vector of length $l_w - 1$, where l_w is the length of a sliding window. Zero components ($F(0)$) were not used because they reflect the overall sum of the elements in sequence, which may dominate causing a false dissimilarity.
4. The resulting vectors are compared using generic distance functions like Euclidean, Cosine or other metrics (Vinga and Almeida, 2003)[4].

[2] The role of sequence representation type was tested on 24 different configurations covering all of the three proposed methods with all the remaining parameters except sequence representation type intact. The root-of-one representation was the most accurate on all the test runs with minimal difference of 4.02%, maximal of 13.4% and average of 10.6%.

[3] FFT and WHT performance comparison was tested analogically on 28 test runs. FFT was more accurate on 27 test runs - minimal difference of -5.32%, maximal of 12.2%, average of 6.19%.

[4] Our implementation uses both Euclidean and Cosine distances, with almost identical performance. Other functions considered (Vinga and

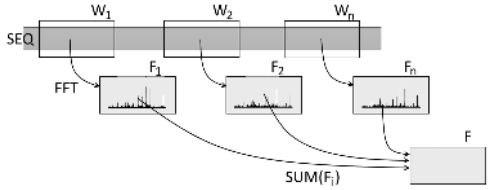


Fig. 4. Sliding window spectral projection process visualization.

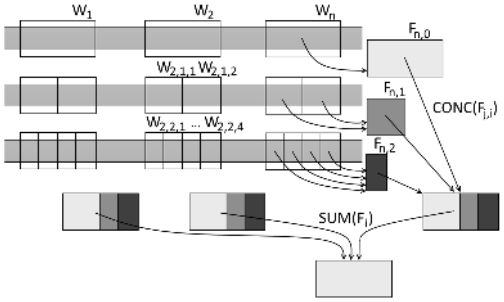


Fig. 5. MRA Sliding window spectral projection process visualization.

Figure 4 shows basic idea of the process. For better readability, the sliding windows are displayed with negative overlap. In practice, better results are acquired using zero to positive overlaps up to 75% of the window length.

The method provides analogy to the A_WFM. In both cases, the problem with genomic changes is handled by considering the sequence as a set of shorter subsequences referred as words or windows. However, the A_WFM is discrete: Sub-result of each word processed is an index, at which the resulting vector will be incremented. Therefore the length of a word is limited to $l_w = \log_4(\frac{l_{seq}}{c})$, where l_{seq} is a sequence length, $c >> 1$ is a constant. Longer words would cause the function, that maps the word to an index, to have a codomain of a higher size implying also longer resulting vectors. These vectors would be very sparse because of lack of values to contain and so the method would be inefficient. On the other hand our method produces numerical vectors of characteristics for words - windows being processed. These vectors need not to be counted but rather summed and so it is possible to use words up to thousands of characters in length. The method, similarly to *[Spaced] Word frequencies*, successfully resolves problems with subsequence transpositions and therefore can be safely used for nuclear sequences.

4.4 MRA approaches

As stated above, inserting/deleting a short part into/from a sequence causes problems for signal processing methods. The problem is remarkable for faster frequencies (with shorter spatial period) as transforms will summarize multiple base function periods into one coefficient. Using smaller windows, the issue is not that serious. However, by definition, windowed transforms are able to compute frequency spectrum for periods only up to the window size, so slower frequencies would be missing. Possible solution is implementing a process of Multiple Resolution Analysis (MRA) (Bey, 2016) that will use smaller windows to compute faster frequencies and longer windows for slower frequencies.

One of known MRA approaches is *Wavelet Transform* or *Haar Wavelet Transform*. This method transforms input vector of length l_{seq} into a multiple resolution signal spectrum containing $\frac{l_{seq}}{2^k}$ coefficients for periods of the length 2^k; $k \in < 0, \log_2(l_{seq}) - 1 >$ and finally one F(0) component. The number of all coefficients converges to l_{seq}, so no information is lost and even an inverse transformation can be computed. However, as stated before, strict spatial information becomes irrelevant due to sequence rearrangements and the only possible way is to sum up all the coefficients corresponding to one period into one element. Because only periods of length of 2^k are processed, the resulting vector has only $\log(l_{seq})$ coefficients. Tests reveal, that $\log(l_{seq})$ number wide vector is too small to provide a good representation of a sequence and therefore the method has much weaker accuracy[5].

To increase the accuracy by computing all coefficients in the possible range $< 0, l_w - 1 >$, not only those reflecting periods of length 2^p, we introduce a MRA modification of the original E_SWM. In this case, each window is processed multiple times with different sub-window sizes.

1. Spectral analysis is computed across the selected window using the whole window length (l_w).
2. Selected window is then splitted into k non-overlapping sub-windows (of size $\frac{l_w}{k}$). Each of the sub-windows is then separately transformed into a signal spectrum. All of the sub-signal spectra are summed into one vector of the length $\frac{l_w}{k}$. Its elements contain spectral analysis of the $\frac{l_w}{k}$ highest coefficients with better precision, so they are placed to corresponding indices $< l_w - \frac{l_w}{k}, l_w)$ in the resulting vector.
3. By executing Step 2, we have computed coefficients in range of $< l_w - \frac{l_w}{k}, l_w)$ with better accuracy. To gather more accurate coefficients in range $< l_w - \frac{l_w}{2k}, l_w)$, we execute Step 2 again with doubled k-s until the desired resolution is reached.

After this, all coefficients in range $< 0, l_w - 1 >$, are computed. The only cost is higher computational complexity, which is still asymptotically better than alignment methods or any pairwise method.
Multiple resolution windows are supposed to be able to identify short motifs in sequences much more precisely and therefore make the original E_SWM method more stable for sequences with frequent insertions or deletions. However, as will be discussed later, the improvement over E_SWM is not significant. This might be due to high amount of data being processed each time. Expected motifs are distributed randomly, with no regard for computational period size and therefore interferences cancel each other as well.

5 Evaluation and Results

Proposed algorithms have been tested and compared to some existing solutions on four different datasets:

Almeida, 2003) retain poorer perfomance with 11.7% up to 21.5% difference.

[5] The method is not stated in final results because of its accuracy is weaker even compared to trivial A_WFM or B_WFM. Compared to E_SWM the minimal difference mixing all test runs and datasets is of 1.38%, maximal of 28.4%, average of 24.2%.

- Mammalian dataset[6] containing 31 well-known animal species, each described by a single 16 kbp[7] mitochondrial sequence.
- Fungi I dataset[8] containing 10 species, 7 mitochondrial sequences of 1-2 kbp per species[9]. Entities (species) are selected from a wider dataset to be as dissimilar as possible[10].
- Fungi II dataset containing 11 entities from the previous source, in this case selected as a compact subtree with similar species[11].
- Fungi III dataset[12] of 10 species contained in the wide Valach's (Valach *et al.*, 2011) dataset[13]. In this case, each species was represented by its full nuclear genome of length $14 - 34Mbp$.

Reference results for Mammals I dataset were obtained by slow alignment methods. In our case it was *MUSCLE (Edgar, 2004)* algorithm hosted by *The European Molecular Biology Laboratory*[14]. The resulting data was a full distance matrix and full species dendrogram constructed on pairwise alignment match counts.

For the Fungi datasets the reference dendrograms are taken from their original publisher Valach *et al.* (2011).

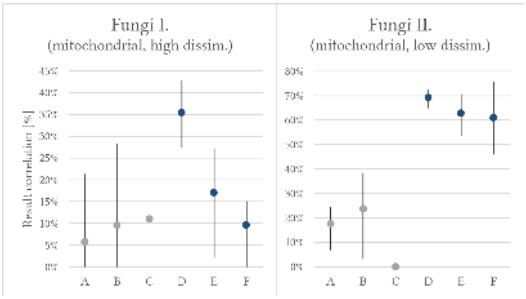


Fig. 6. Experimental results accuracy using Fungi I and II data. Our methods are referred as D, E, F.

5.1 Evaluation Metrics

Our goal is to evaluate similarity between the reference dendrogram and the one produced by clustering algorithm based on the distance matrix computed using the proposed spectral transform approaches. The reason for comparing the dendrograms and not distance matrices is that dendrogram reflects both distance matrix values and clustering relevance obtained by the related matrix.

There are various metrics for evaluation of a tree similarity, e.g. metrics based on graph topology such as edit distance. These, however, would not reflect connection edge lengths - metric distances between species (entities), because connecting edge lengths are not preserved in topological information.

Our approach was to compute a new pairwise distance matrix, based on dendrogram tree traversal. The matrix can be computed with two possible variants. In the first (metric) version, the distances between entities are computed as the length of the unique traversal path across the tree, summing the lengths of connecting edges. The second (discrete) topological version sets the length of each edge to one and so reflects the number of connecting edges. This approach was used in experiments with Fungi III nuclear dataset where the reference metric values were computed using mitochondrial sequences, and therefore these metric values are irrelevant. This variant does not profit from edge length information, however was selected to be consistent with other experiments.

As the lengths of edges in the final resulting dendrogram vary in dependence on the original distance matrix scale and clustering method used, the metric version needs to normalize the resulting distance matrices to equal mean and standard deviation (z-index). Final accuracy score for a configuration is then computed as absolute scalar difference of resulting dendrogram traversal matrix and reference dendrogram traversal matrix, what can be expressed as $score = \sum_{i=1}^{m} \sum_{j=1}^{m} |resm_{i,j} - refm_{i,j}|$ where m is number of entities and $resm, refm$ are matrices as described above. Presented percentual values referred as correlation reflect position of the computed $score$ against reference result ($resm = refm$; $score = 0$) placed at 100% mark and $score$ computed for $resm$ with uniform random distribution at 0% mark.

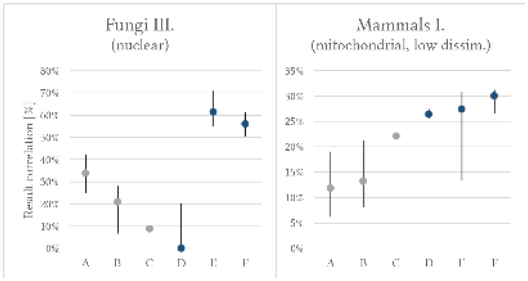


Fig. 7. Experimantal accuracy using Fungi III and Mammals I data. Our methods are referred as D, E, F.

5.2 Configurations and settings

During the pre-testing phase we selected configurations that have achieved the best accuracy. These configurations were then used for the testing phase:

- A_WFM with the word lengths $\{4, 6, 8\}$.
- B_SWFM configured to use the words of lengths $\{10, 16, 24\}$ with $\{4, 6, 8\}$ *match* positions.
- C_YMM[15].

[6] Dataset included in supplementary materials of work by Yin et al. (Hoang *et al.*, 2015; Yin *et al.*, 2014; Yin and Wang, 2014).

[7] To be able to compare results with faulty C_YMM, all input sequences needed to be truncated to the same length of 16000 bp.

[8] Dataset and results contained in supplementary material of work by Valach et al. (Valach *et al.*, 2011).

[9] Genes used in Fungi I and Fungi II datasets: ATP6, ATP8, ATP9, COB, COX1, COX2, COX3

[10] Species used in Fungi I dataset: aspNig, canZem, yarLip, canSub, canAlb, canNer, canFri, canPar, sacPas, sacCer, sacSer.

[11] Species used in Fungi II dataset: canTro, canSoj, canVis, canFri, canNer, canDub, canAlb, canMal, debHan, picFar.

[12] Data obtained directly from NCBI database (https://www.ncbi.nlm.nih.gov/).

[13] Species used in Fungi III dataset: aspNig, debHan, canMal, canAlb, canDub, canSoj, canTro, canPar, sacPas, sacCer

[14] The European Molecular Biology Laboratory. MUSCLE main page: http://www.ebi.ac.uk/Tools/msa/muscle/

[15] No configurable parameters.

- D_DCM with FFT, root-of-one representation and $\{5, 10, 20\}\%$ of dominant coefficients.
- E_SWM using the FFT, root-of-one representation and window sizes in range $< min(1024, l_{seq}/16), min(4096, l_{seq}/4) >$ and overlaps in range $< 0, 75 > \%$[16].
- F_MSWM with sizes and overlaps configured as E_SWM and $k = \{4, 8\}$.

5.3 Evaluation of experiments

Our experiments prove, that method accuracy is affected mainly by two factors. They are: (I.) The average entity dissimilarity in dataset. (II.) The inner character of sequences (mitochondrial or nuclear). Standing on 195 test runs we have found the following:

- As expected, methods using the whole sequence for analysis (C_YMM and D_DCM) are not processing nuclear sequences correctly, therefore reach zero or even sub-zero correlations of the result with the reference result. On the other hand, they perform very well on mitochondrial sequences of high dissimilarity, where they are able to recover significant differences being not restricted by a limited context provided by a short window .
- Proposed E_SWM and F_MSWM are the only two methods to work well with nuclear sequences. The accuracy of these methods on mitochondrial sequences is also above average but D_DCM is more stable and accurate on this type of data.
- Results on datasets Mammals I and Fungi II with similar characteristics are, as expected, similar except C_YMM. As stated above, this method contains erroneous normalization and requires equal input sequence lengths, which we were not able to provide on Fungi I and II data without biasing other test runs.
- Computational complexity and running times of windowed methods (A_WFM, B_SWFM, E_SWM, F_MSWM) are asymptotically equal. Test running times for various configurations of each method averaged $1937, 17693, 15645, 42835ms$, respectively, on Fungi III data. We notice $\log_2(k) + 1$ complexity increase of F_MSWM compared to E_SWM, caused by each window being processed $\log_2(k) + 1$ times (this case $k = 4$). The complexity of non-windowed methods is higher, leading to approximately $200s$ running time in average, however with the only sensible use on short mitochondrial sequences, what is irrelevant.

Enclosed *min-max-average* figures (Fig. 6, Fig. 7) - display minimal, maximal and arithmetic average of all the accuracy values for configurations as defined above grouped by sequence comparison method used.

6 Conclusion

We propose three novel domain-specific methods for evaluation of pairwise similarity of DNA sequences. The methods are based on spectral transforms and perform up to three times better than other known methods. This is due to:

(I.) Considering various domain-specific problems stated above as *short insertion problem* and *transposition problem*.

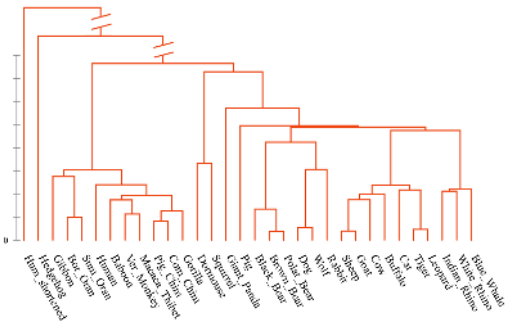


Fig. 8. Final dendrogram on Mammals I data constructed by C_YMM (22.1% correlation). Results differ from the original paper because of input sequence truncation to a longest common length. Before, uneven lengths helped to arrange the entities into the expected structure.

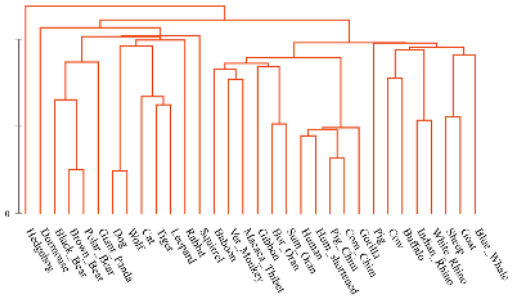


Fig. 9. Final dendrogram on Mammals I data constructed by F_MSWM (31.2% correlation).

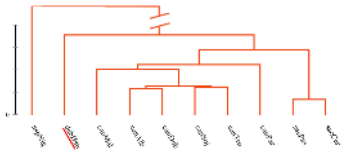


Fig. 10. Final dednrogram on Fungi III (raw nuclear) data constructed by E_SWM (62.4% correlation) with a single false outlier.

(II.) Considering various parameters such as spectral transform type, DNA sequence representation type, and window lengths.

Based on theoretical properties proven by experiments we encourage to use the proposed E_SWM or F_MSWM methods (configured as stated above) for comparison of nuclear sequences and D_DCM for any mitochondrial data comparison.

Acknowledgement

A very special gratitude goes out to Assoc. Prof. Bronislava Brejova, PhD. for providing knowledge in the fields related to biology and bioinformatics

[16] Up to -125% overlap on nuclear sequence data (window gap equal to 125% of window length).

[16] For creating the displayed dendrograms, the Mammals I dataset was enriched by a shortened 13kbp long Human sequence to show wrong sequence normalization in C_YMM. The original tests and accuracy computations did not use this artificial entity.

and for her valuable remarks helping us to improve the quality of the paper. This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG 1/0752/14.

References

Bey, N. Y. (2016). Multi-resolution fourier analysis: achieved high resolutions with suppressed finite observation effects. *Signal, Image and Video Processing*, **10**(4), 711–718.

Borrayo, E., Mendizabal-Ruiz, E. G., Vélez-Pérez, H., Romo-Vázquez, R., Mendizabal, A. P., and Morales, J. A. (2014). Genomic signal processing methods for computation of Alignment-Free distances from DNA sequences. *PLoS ONE*, **9**(11), e110954+.

Cooley, J. W. and Tukey, J. W. (1965). An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, **19**(90), 297–301.

Edgar, R. C. (2004). Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, **5**(1), 113.

Goel, S., Kaur, G., and Tomar, P. (2015). Performance analysis of welch and blackman nuttall window for noise reduction of ecg. In *2015 International Conference on Signal Processing, Computing and Control (ISPCC)*, pages 87–91.

Gronau, I. and Moran, S. (2007). Optimal implementations of upgma and other common clustering algorithms. *Inf. Process. Lett.*, **104**(6), 205–210.

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, **66**(1), 51–83.

Haubold, B. (2014). Alignment-free phylogenetics and population genetics. *Briefings in Bioinformatics*, **15**(3), 407–418.

Hoang, T., Yin, C., Zheng, H., Yu, C., He, R. L., and Yau, S. S.-T. (2015). A new method to cluster DNA sequences using Fourier power spectrum. *Journal of Theoretical Biology*, **372**, 135 – 145.

Kantorovitz, M. R., Robinson, G. E., and Sinha, S. (2007). A statistical method for alignment-free comparison of regulatory sequences. In *Proceedings 15th International Conference on Intelligent Systems for Molecular Biology (ISMB) & 6th European Conference on Computational Biology (ECCB), Vienna, Austria, July 21-25, 2007*, pages 249–255.

Katoh, K. and Toh, H. (2008). Recent developments in the MAFFT multiple sequence alignment program. *Briefings in Bioinformatics*, **9**(4), 286–298.

Leimeister, C., Boden, M., Horwege, S., Lindner, S., and Morgenstern, B. (2014). Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics*, **30**(14), 1991–1999.

Leimeister, C.-A. and Morgenstern, B. (2014). KMACS: The k-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics*, **30**(14), 2000–2008.

Ouyang, W. and Cham, W. (2010). Fast algorithm for Walsh Hadamard transform on sliding windows. *IEEE Trans. Pattern Anal. Mach. Intell.*, **32**(1), 165–171.

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, **4**(4), 406–425.

Schwende, I. and Pham, T. D. (2014). Pattern recognition and probabilistic measures in alignment-free sequence analysis. *Briefings in Bioinformatics*, **15**(3), 354–368.

Sibson, R. (1973). Slink: An optimally efficient algorithm for the single-link cluster method. *Comput. J.*, **16**(1), 30–34.

Song, K., Ren, J., Reinert, G., Deng, M., Waterman, M. S., and Sun, F. (2013). New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Briefings in Bioinformatics*, **15**(3), 343.

Valach, M., Farkas, Z., Fricova, D., Kovac, J., Brejova, B., Vinar, T., Pfeiffer, I., Kucsera, J., Tomaska, L., Lang, B. F., and Nosek, J. (2011). Evolution of linear chromosomes and multipartite genomes in yeast mitochondria. *Nucleic Acids Research*, **39**(10), 4202–4219.

Vinga, S. and Almeida, J. (2003). Alignment-free sequence comparison-a review. *Bioinformatics (Oxford, England)*, **19**(4), 513–523.

Yin, C. (2016). Identification of repeats in DNA sequences using nucleotide distribution uniformity. *CoRR*, **abs/1608.00567**.

Yin, C. and Wang, J. (2014). A novel method for comparative analysis of DNA sequences by ramanujan-fourier transform. *CoRR*, **abs/1403.1523**.

Yin, C., Chen, Y., and Yau, S. S.-T. S. (2014). A measure of DNA sequence similarity by Fourier Transform with applications on hierarchical clustering. *Journal of theoretical biology*, **359**, 18–28.

STU FIIT
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

NOVEL DNA ALLIGNMENT-FREE COMPARISON
BASED ON SIGNAL PROCESSING APPROACHES

Tomáš Farkaš
supervisor: Mária Lucká

ABSTRACT

Computing similarity between two nucleotide sequences is a non-trivial task because of features specific for the sequences. Current methods are either computationally expensive or purely and so less accurate.
We propose novel methods based on signal processing transforms designed for overcoming sequence specificities while preserving computational modesty. The experiments reveal, that the novel methods are up to three times better compared to current alignment-free methods in the means of accuracy, while they are equally computationally inexpensive.

INTRODUCTION

Hierarchical clustering
- Creating a tree diagram of species to show mutual relations
- Need to know their similarity - based only on DNA sequence

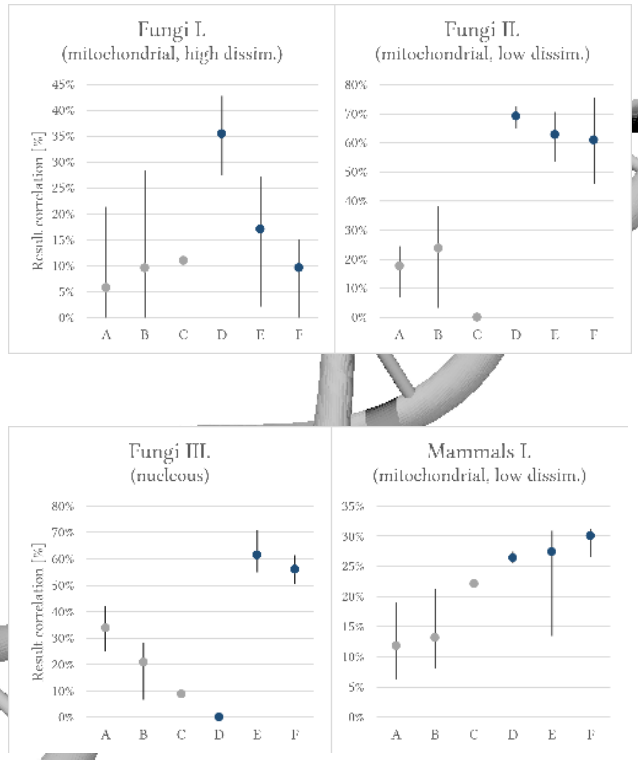
DNA sequence specifities
- Subsequence insertions, deletions, transpositions ...
- Inaccurate spatial information

CURRENT METHODS

Full sequence allignment
- Very computationally expensive
- Not applicable on raw data

Statistic methods
- Word Frequencies (A)
- Spaced Word Frequencies (B)
- Inaccurate

Early FFT methods (C)



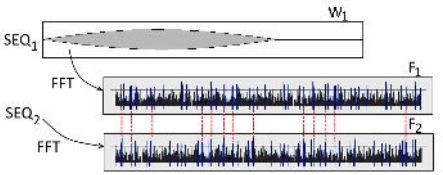
REFERENCES

- Yin, C., Chen, Y., and Yau, S. S.-T. S. (2014). A measure of DNA sequence similarity by Fourier Transform with applications on hierarchical clustering. Journal of theoretical biology, 359, 18–28
- Song, K., Ren, J., Reinert, G., Deng, M., Waterman, M. S., and Sun, F. (2013). New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. Briefings in Bioinformatics, 15(3), 343.

NEW METHODS

DOMINANT COEFFICIENT METHOD (D)

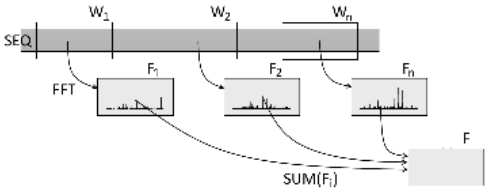
1. Compute spectrum of whole sequences
   - Smoothed using BNW
   - Padded to equal length
2. Find number of pairwise matches
   - Positions of coefficients with value above a threshold



SLIDING WINDOW METHOD (E,F)

Windowed processing
- No problem with rearranged sequence parts

1. Split a sequence into windows
2. Compute spectra of windows and sum them
3. Compare entities using resulting sum-spectrum vectors



ACKNOWLEDGEMENT
This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG 1/0752/14.