

# Sagedasti esinevate sõnemustrite otsimine ja algoritm Teiresias

Ireen Meho  
Arvutiteaduse instituut, Tartu Ülikool  
ireenme@ut.ee

Andmekaevandamise uurimisseminar MTAT.03.169.  
Arvutiteaduse instituut, Tartu Ülikool  
Detsember 2003, lk. 101–111

## Kokkuvõte

Käesolevas artiklis tutvustame lühidalt mustrite otsimise meetodeid ning lähemalt kirjeldame algoritmi Teiresias, mis otsib antud bioloogiliste sekventside hulgas esinevaid fikseeritud pikkusega mustreid. Algoritm leiab kõik mustrid, mis esinevad sisendhulgas vähemalt etteantud arv korda. Seejuures väljastatakse ainult sellised mustrid, mis täidavad maksimaalsuse tingimust, st. neid ei saa muuta täpsemaks ilma, et esinemise sagedus väheneks. Tegemist on väga tõhusa otsimismeetodiga, kuna algoritmi töö käigus ei vaadelda läbi kogu lahendiruumi, vaid ainult potentsiaalseid lahendeid (st piisava esinemissagedusega maksimaalseid mustreid). Lisaks tutvume mõningate nimetatud algoritmi rakendustega, mis on saadaval IBM'i bioinformaatika ja mustrite otsimise grupi (*IBM's Bioinformatics and Pattern Discovery Group*) veebiserveris.

# 1 Sissejuhatus

Viimase kümnendi jooksul on huvi suurtest andmehulkadest sageli (või vastupidi, väga harva) esinevate sõnede otsimise vastu kõvasti kasvanud. Tähtis roll on siin geeniuuringute arengul, kuna bioloogilistest sekventsistest koosnevaist hulkadest olulise informatsiooni leidmiseks on vaja efektiivseid vahendeid. Sarnaste sõnede ehk mustrite otsimise meetodid on üks võimalus selle probleemi lahendamiseks. Loomulikult ei ole bioloogia ja geeniuuringud mustrite otsimise meetodite ainus rakendusvaldkond, huvi pakkuda võivaid regulaarsusi saab otsida mitmesugustest tekstilisel kujul esitatavatest andmetest, nagu näiteks inimkõne, muusika, arvud jne.

Mustri headus on olnud tähtis probleem alates andmekaeve uuringute algusest. Erinevatel kasutajatel on sageli vaja erinevate omadustega mustreid ja suur hulk leitud mustreist võib osutada ebahuvitavaiks. Kõigi mustrite hulgast sobivate välja filtreerimiseks on pakutud mitmeid faktoreid, mis võiksid mustri huvitavust näidata [3]. Näiteks sagedus, sarnasus, statistiline olulisus, lihtsus jmt — kõiki neid nimetatakse kokkuvõtvalt objektiivseteks omadusteks. Lisaks neile on omadused, mida nimetatakse subjektiivseteks: üllatavus — muster on huvitav, kui see on kasutajale tundmatu või selle esinemine on ootamatu — ja kasutatavus (ingl *actionability*) — muster on huvitav, kui sellega on võimalik midagi kasulikku teha. Need kaks ei ole teineteist välis- tavad. On täheldatud, et kuigi objektiivsed omadused on mitmeti kasulikud, ei piisa neist alati mustri headuse hindamiseks [3].

Mustrite otsimiseks peab olema täidetud kolm tingimust. Määratud peavad olema

1. mustri esitamise keel, millega pannakse paika leitavate mustrite kuju;
2. mustri "headuse" funktsioon. Selleks võib olla näiteks mustri esinemise sagedus või tõenäosus, informatsiooni sisalduvus jmt.
3. otsimisalgoritm, millega kirjeldatakse kõikvõimalike mustrite hulgast meid huvitavate mustrite otsimise viis.

Otsimisalgoritme on erinevate mustrite otsimisega seotud probleemide lahendamiseks loodud mitmeid, igauks neist läheneb asjale veidi erineva nurga alt. Käesolevas töös tutvustamegi lühidalt mõningaid otsimise meetodeid, pikemalt peatume algoritmil Teiresias, mis parema tulemuse saavutamiseks kombineerib lihtsaid otsimismeetodeid ning lisaks rakendab efektiivset pügamismeetodit, et vähendada algoritmi tööaega. Põhjaliku vaatluse alt jäävad välja Teiresias'e edasiarendused keerukamate mustrite ja veelgi tõhusama rakendamise suunal.

## 2 Ülevaade mustrite otsimise meetoditest

### 2.1 Mõisted

Tähistame sümboliga  $\Sigma$  sisendandmetes esineda võivatest tähetedest koosnevat tähestikku. Näiteks DNA puhul on tähestikuks  $\Sigma = \{A, C, G, T\}$ . Reeglina saab mustri otsimise algoritme rakendada suvalisel lõplikul tähestikul.

Mustri mõistele lähenevad erinevad algoritmid erinevalt. Kõige üldisemal tasemel võiks mustrid jagada deterministlikeks ja tõenäosuslikeks [1]. Esimest võib vaadelda tähestiku tähtedest ja punktidest (tähistab sõne positsiooni, milles võib esineda üks suvaline tähestiku täht) ja/või tärnidest (tähistab sõne lünka, milles võib esineda suvaline arv suvalisi tähestiku tähti) koosneva regulaaravaldisena, mis esineb otsisõnes või mitte. Teisel juhul antakse igale mustriale tõenäosus, millega ta vastavas sõnes esineda võib. Mida kõrgem on tõenäosus, seda paremini sobitub muster teksti.

Lihtsaim tõenäosuslike mustrite esitusviis on kaalumatriks (position-weight matrix), mille puhul esitatakse muster tabelina seades igale tähele vastavusse antud positsioonis esinemise sageduse. Kaalumatriksit (vahel nimetatakse seda ka profiiliks) üldistavad varjatud Markovi mudelid (Hidden Markov Models, HMMs), mis võimaldavad lisaks teksti ja mustri sümbolite erinevustele kujutada ka sümboli lisamist mustrisse ja eemaldamist mustrist.

Selles töös vaatleme mustreid, mis algavad ja lõpevad tähestiku  $\Sigma$  tähega ning sisaldavad suvalist kombinatsiooni tähestiku tähtedest ja punktidest [5], ehk sõnesid kujul

$$\Sigma(\Sigma\{'.\})^*\Sigma$$

Kuna tegemist on regulaaravaldisega, siis öeldakse, et muster  $P$  määrab keele  $G(P)$ . Keele elemendid saadakse mustrist  $P$  kõigi punktide asendamisel tähestiku  $\Sigma$  mingi tähega. Näiteks on muster järgnev sõne:

$$'A.T.A'$$

ning selle mustri poolt määratud keele mõned elemendid on

$$ATTCA, ACTCA, AGTCA$$

Mustri  $P$  osasõnet, mis on samuti muster, nimetatakse mustri  $P$  osamustriks.

Tähestiku  $\Sigma$  tähtedest koosnev sõne  $s$  ühtib (*match*) antud mustriga  $P$ , kui  $s$  sisaldab vähemalt ühte osasõnet, mis kuulub keelde  $G(P)$  [5]. Näiteks järgmise hulga elemendid ühtivad mustriga 'A.T.A':

$$S = \{ATACTTAC, ACAGTCA, AATGATT\}$$

Mustri kokkulangevust sõnega märgib selles näites paksus kirjas sõne osa.

Mustri esinemiste meelespidamiseks defineerime uue andmestruktuuri järgmiselt: mustri  $P$  algusloendiks (ingl *offset list*) sekventsise hulga  $S = \{s_1, s_2, \dots, s_n\}$  jaoks nimetame hulka

$$L_S(P) = \{(i, j) \mid \text{sekvents } s_i \text{ ühtib mustriga } P \text{ alates positsioonist } j\}$$

Näiteks eeltoodud hulga  $S$  jaoks on mustri 'A.T.A' algusloend selline:

$$L_S('A.T.A') = \{(1, 3), (2, 3), (3, 1)\}$$

Öeldakse, et muster  $P'$  on täpsem kui muster  $P$ , kui  $P'$  on tuletatav mustrist  $P$ , asendades vähemalt ühe punkti tähega või lisades suvalise tähtedest ja punktidest koosneva sõne  $P$  algusse ja/või lõppu. Mustrist 'A.T.A' on täpsemad näiteks mustrid

$$'AGT.A', 'CTA.TAAG', 'CTA.T.A..G'$$

Saab näidata, et kui muster  $P'$  on täpsem kui muster  $P$ , siis iga sekventsise hulga  $S$  korral

$$|L_S(P')| \leq |L_S(P)|$$

Maksimaalseks nimetame mustrit  $P$ , mille jaoks sekventsise hulgal  $S$  ei leidu mustrit  $P'$ , mis oleks täpsem kui  $P$  ning seejuures kehtiks  $|L_S(P)| = |L_S(P')|$ . Teisisõnu, kui  $P$  on maksimaalne muster, siis ei saa teda täpsemaks muuta ilma, et algusloend samal ajal väheneks.

## 2.2 Meetodid

Sageli on ülesanded, mille lahendamiseks mustrite otsingut kasutatakse, üpris arvutusmahukad ning ei ole võimalik leida kiiret algoritmi parima lahendi saamiseks. Seepärast kasutavad paljud meetodid nn. põhjalikku otsingut (ingl *exhaustive search*). Lihtsaim sellistest lähenemistest on kõikvõimalike mustrite loetlemine (ingl *enumeration*) ja neile mingi esinemisskoori vastavusse seadmine. Seejärel on võimalik väljastada maksimaalse skooriga mustrid või mustrid, mille skoor on mingist väärtusest suurem [1]. Mustri avastamiseks on sageli kasutusel joondamise tehnika, kus seatakse sõned paarikaupa kohakuti ja võrreldakse vastavaid sümboleid. Järgmisel sammul nihutatakse ühte sõnedest ühe koha võrra ja korratakse võrdlemise protseduuri.

Täpse otsimise mõistele vastandub ligikaudse otsimise mõiste, mis hõlmab endas otsimist  $k$  erinevuse (ingl *mismatch*) korral ning otsimist  $k$  teisenduse (ingl *error*) korral. Neist esimese puhul on ülesandeks leida kõik sellised mustrid  $P$ , milles on maksimaalselt  $k$  erinevust  $P$  iga esinemise korral tekstis  $s$

(lubatud teisendus on asendamine). Teine juhtum tähendab, et tuleb leida kõik osasõned, mille saamiseks tuleb teha maksimaalselt  $k$  teisendust. Lubatud teisendusteks on tähe lisamine, eemaldamine, asendamine. Ühest sõnest teise saamiseks vajalike lisamiste, eemaldamiste ja asendamiste vähimat arvu nimetatakse teisenduskauguseks. Sellega analoogiline mõõt on Hammingi kaugus, mis on võrdne mustri saamiseks tehtud asendamiste vähima arvuga.

Ligikaudse otsimise meetodite aluseks on teisenduskauguse või Hammingi kauguse arvutamine. See tähendab, et vaadeldav muster pakub huvi, kui tema saamiseks tehtavate teisenduste arv on väiksem (või võrdne) kui ette antud kaugus. Mõlema kauguse arvutamiseks on võimalik kasutada dünaamilise programmeerimise ja diagonaalimeetodit [4]. Mõned ligikaudse otsingu meetodid on näiteks otsimine sufiksipuust või -triest, aja painutamine, samuti nt optimaalse joonduse leidmine dünaamilist programmeerimist kasutades.

Keerulisemate mustrite korral võib juhtuda, et tuleb kasutada mingeid iteratiivseid heuristikuid, mis võivad koonduda lokaalsesse maksimumi. Sellisel juhul võib algoritm mitte leida parimat võimalikku mustrit, vaid leiab peaaegu parima mustri. Näide sellisest lähenemisest on *Gibbs Sampling*, meetod, mis igal tsükklisammul leiab juhusliku  $W$ -sümboliliste osasõnede hulga, kus on iga sõne kohta täpselt üks osasõne ja arvutab neile vastavad kaalu-maatriksid. Selliselt valitud osasõnede korral võib parimat skoori andev hulk leidmata jääda. Samas on vaatlusalune meetod kiire, mistõttu on ta paljude rakenduste puhul sobiv lahendus.

Iteratiivseid meetodeid on teisigi. Tavaliselt alustatakse mingi mustriga, otsides selle kõiki esinemisi. Vastavalt leitud esinemistele genereeritakse muster, mis neisse positsioonidesse kõige paremini sobib ja alustatakse uut ringi uue mustriga. Sedaviisi tehakse, kuni mustrit ei saa enam parandada. Protsess on deterministlik ja, nagu teiste iteratiivsete meetodite puhulgi, ei garanteeri parima lahenduse leidumist [1].

Varem kirjeldatud mustrite loetlemise meetod osutub efektiivseks vaid lühemate mustrite otsimisel, kuid olulist tähendust omavad sageli just pikemad mustrid. Üks võimalik lahendus sellele probleemile on lühikeste mustrite kombineerimine üheks pikaks. Järgnevas tutvustame algoritmi, mis rakendab just seda lähenemist.

Teiresias on täpse otsingu algoritm, mis leiab ja väljastab kõik sisendhulgas piisavalt sageli esinevad maksimaalsed fikseeritud pikkusega mustrid. Sisuliselt on tegemist sügavuti otsingu puuga, mille läbimisel vaadeldakse ainult potentsiaalselt nõuetele vastavaid lahendeid andvaid harusid. Seega ei joondata sekventse paarikaupa ega loetleta üles kõikvõimalikke mustreid, mistõttu algoritmi töökiirus on teiste põhjalikul otsingul baseeruvate meetoditega võrreldes suurem [5]. See omakorda võimaldab otsida pikemaid ja keerulisemaid mustreid.

Algoritm Teiresias jaguneb kaheks etapiks: läbivaatus (ingl *scanning*) ja ühendamine (ingl *convolution*). Esimesel etapil võrreldakse sisendhulga  $S$  elemente ja leitakse kõik vähemalt  $K$  erinevas sekventsisis esinevad elementaarmustrid. Saadud elementaarmustrite hulk  $S_E$  on sisendiks teisele etapile, mil mustreid omavahel kombineerides genereeritakse järk-järgult kõik maksimaalsed mustrid. Hulga  $S_E$  iga elementaarmustrit laiendatakse esmalt suffiksi ja seejärel prefiksi suunas ning igal sammul kontrollitakse tulemuse maksimaalsust ja sekventsides esinemise sagedust. Kui laiendamisel saadud muster ei ole seni leitudtega võrreldes maksimaalne, jäetakse ta edasise vaatluse alt välja. Sama kehtib mustrite kohta, mis esinevad vähem kui  $K$  erinevas sisendhulga  $S$  sekventsisis. Nõutava mustrite esinemise sageduse ja pikkuse annab ette kasutaja.

Sisendhulga  $S = \{s_1, s_2, \dots, s_n\}$  korral on Teiresias'e läbivaatusfaasi keerukus

$$O(mW^L)$$

ning ühendamisfaasi keerukus halvimal juhul

$$O\left(W^L m \log m + W \left(Cm + t_H \sum_{\max P} rc(P)\right)\right),$$

kus  $m$  on tähtede arv hulgas  $S$  ehk  $m = \sum |s_i|$ ,  $W$  ja  $L$  on konstantsed parameetrid, mis märgivad vastavalt elementaarmustri vähimat pikkust ja tähtede arvu elementaarmustris,  $C$  on leitud maksimaalsete mustrite keskmine arv,  $t_H$  paisktabeli kirje leidmisele kuluv aeg,  $P$  genereeritav maksimaalne muster ja  $rc(P)$  tähtede arv mustris  $P$  [6].

Käesolevas töös vaadeldavat algoritmi versiooni on mitmes erinevas suunas täiendatud ja edasi arendatud, kuna see ei lahenda näiteks varieeruva pikkusega mustrite otsimise probleemi [7] ega võimalda kasutada märgiklasse. Samuti eksisteerib võimalus, et programm töötab eksponentsiaalse ajaga, kuna algoritmi soorituse aeg sõltub sisendhulga sekventsides sarnasuse määrist [5, 6], artikli [1] andmetel on olemas lahendus juba ka sellele probleemile.

### 3 Teiresias: realisatsioon

Mõistmaks Teiresias'e algoritmi paremini, tuleb eelnevalt defineerida mõned mõisted. Mustrit  $P$  nimetatakse  $\langle L, W \rangle$  mustriks (kus  $L \leq W$ ), kui mustri  $P$  iga osamuster, milles on vähemalt  $L$  tähestiku  $\Sigma$  tähte, on pikkusega vähemalt  $W$  sümbolit. Sümboliteks võivad olla nii tähed, kui ka suvalist tähte tähistav punkt. Elementaarmuster on  $\langle L, W \rangle$  muster, milles on tähti täpselt  $L$  ning punkte seega vähemalt  $W - L$ . Olgu näiteks  $L = 3$  ja  $W = 4$ .

Siis mõned sekventsides

TACTTAC, ACATAGT, ATACTTA

esinevad  $\langle L, W \rangle$  elementaarmustrid on järgmised:

'AC.T', 'A..TA', 'AC..A', 'C.TA'

Olgu  $prefix(P)$  vähemalt  $L$ -tähelise mustri  $P$  osamuster, mis koosneb täpselt  $L - 1$  tähest ja tähistab mustri  $P$  prefiksit. Analoogiliselt, märki-  
gu  $suffix(P)$  mustri  $P$  ( $L - 1$ )-tähelist sufiksit. Näiteks  $L = 5$  korral

$$\begin{aligned} prefix('CTA.T.A..G') &= 'CTA.T' \\ suffix('CTA.T.A..G') &= 'A.T.A..G' \end{aligned}$$

Olgu  $P, Q$  mustrid, kummaski vähemalt  $L$  tähte. Siis mustrite  $P$  ja  $Q$  ühen-  
diks (ingl *convolution*) nimetatakse sellist mustrit  $R$ , et

$$R = P \oplus Q = \begin{cases} PQ', & \text{kui } suffix(P) = prefix(Q) \\ \emptyset, & \text{muidu} \end{cases}$$

kus  $Q = prefix(Q)Q'$  ehk  $Q'$  on see osa mustrist  $Q$ , mis jääb järele peale  
prefiksi eemaldamist ja  $\emptyset$  tähistab tühja sõnet. Näiteks  $L = 3$  korral

$$\begin{aligned} 'AC.T' \oplus 'C.TAG' &= 'AC.TAG' \\ 'A..TA' \oplus 'AC..A' &= \emptyset \end{aligned}$$

Mustreid  $P, Q$  nimetatakse ühenduvaiks (ingl *convolvable*), kui  $P \oplus Q \neq \emptyset$ .  
Kui kaks mustrit  $P, Q$  on ühenduvad, siis mustri  $R = P \oplus Q$  algusloend  $L_S(R)$   
on hulga  $L_S(P)$  alamhulk, mis defineeritakse järgmiselt:

$$L_S(R) = \{(i, j) \in L_S(P) \mid \exists (i, k) \in L_S(Q) \text{ nii, et } k - j = |P| - |suffix(P)|\},$$

kus  $|P|$  tähistab sümbolite (nii tähtede kui punktide) arvu mustris  $P$ .

Olgu  $P, Q$  kaks suvalist mustrit. Otsustamaks, kumb muster on prefik-  
si järgi väiksem, kasutatakse järgnevat protseduuri. Kõigepealt joondatak-  
se mustrid nii, et nende vasakpoolseimad tähed on kohakuti. Näiteks, kui  
 $P = 'CTA.TAAG'$  ja  $Q = 'CTA.T.A..G'$ , siis näeks joondus välja selline:

P: CTA.TAAG  
Q: CTA.T.A..G

Seejärel võrreldakse joonduse kõiki veergusid vasakult paremale senikaua,  
kuni veeru üks üks sümboleist on täht ja teine punkt. Eelnevat näidet jätkak-  
tes:

P: CTA.TAAG  
 ||||x  
 Q: CTA.T.A..G

Kui täht on mustris  $P$ , nagu antud näites, siis öeldakse, et  $P$  on prefiksi järgi väiksem kui  $Q$  ja kirjutatakse  $P <_{pf} Q$ .

Analoogiliselt saab kindlaks teha, kas muster  $P$  on suffiksi järgi väiksem kui  $Q$  (tähistatakse  $P <_{sf} Q$ ). Selleks tehakse joondus mustrite parempoolseima tähe järgi:

P: CTA.TAAG  
 Q: CTA.T.A..G

ja veergusid võrreldakse paremalt vasakule:

P: CTA.TAAG  
 x|  
 Q: CTA.T.A..G

Nüüd, kus meil on olemas kõik vajalikud eelteadmised, on võimalik kirjeldada algoritmi, mille ülesandeks on antud sekventsides hulga  $S = \{s_1, \dots, s_n\}$  ja parameetrite  $L, W, K$  korral leida kõik maksimaalsed  $\langle L, W \rangle$  mustrid, mis esinevad vähemalt  $K$  erinevas hulga  $S$  sekventsides. Eeldatakse, et väärtused parameetritele  $L, W, K$  sisestab kasutaja.

Algoritmi sisendiks on sekventsides hulk  $S$  ja parameetrite  $K$  (sagedus),  $W$  (mustri pikkus) ja  $L$  (tähtede arv elementaarmustris) kasutaja poolt sisestatud väärtused. Mõistlik on valida  $L \geq 3$ , kuna väiksemate väärtuste puhul on kasutatavad prefiks ja suffiks liiga väikesed ning mustrite laiendamine muutub ebaefektiivseks.

Nagu varem öeldud, on algoritmil kaks etappi, läbivaatuse ja ühendamise etapp. Esimesel etapil võrreldakse sisendhulga  $S$  sekventse sümbolhaaval, otsides vähemalt  $K$  korda esinevaid elementaarmustreid. Mustrit kujutava andmestruktuurina kasutatakse paari  $(P, L_S(P))$ , kus  $P$  on muster sõne kujul ja  $L_S(P)$  on mustri vastav järjestatud algusloend [6]. Mustri esinemise piisavat sagedust saab tagada, kontrollides mustri algusloendi suurust.

Sobiva sagedusega mustrite hulk  $S_E$  saab sisendiks teisele etapile. Kõigepealt järjestatakse sisendhulk prefiksi järgi ning genereeritakse prefiksiste (suffiksiste) hulgad  $DirP$  (vastavalt  $DirS$ ) iga sõne  $w$  jaoks. Need hulgad on realiseeritud tasakaalustatud puudena, mille igas tipus on mittetühi loend sellistest mustritest  $P$ , et  $prefix(P) = w$  (või vastavalt  $suffix(P) = w$ ). Iga loend on vastava osalise järjestusega järjestatud. Nii moodustatud hulkasid on hiljem vaja selleks, et kergemini leida vaadeldava mustri ühendamiseks sobivaid mustreid.

Kui eeltöötlus tehtud, eemaldatakse hulgast  $S_E$  prefiksi järgi väikseim elementaarmuster ja pannakse pinusse. Seejärel proovitakse esialgset mustrit tsükliliselt laiendada, lisades igal sammul pinusse uusi mustreid. Pinu elemendid järjestatakse samuti prefiksi järgi, st kui  $P <_{pf} Q$ , siis muster  $P$  on pinu tipule lähemal kui  $Q$ . Sellega tagatakse, et kõik mustrid genereeritakse ja iga maksimaalne muster genereeritakse enne, kui ükski mittemaksimaalne muster.

Algoritm töötleb alati mustrit, mis asub pinu tipus; seda nimetatakse käesolevaks tipuks. Kõigepealt laiendatakse käesolevat tippu  $P$  sufiksi järgi paremale. Selleks otsitakse hulgast  $DirS$  mustrit  $Q$ , mis on mustriga  $P$  ühenduv ehk  $Q \oplus P \neq \emptyset$ . Kui sobivaid mustreid  $Q$  on rohkem kui üks, siis ühendatakse muster  $P$  ükshaaval iga mustriga  $Q_i$ , alates väikseimast. Täpsemalt, kui  $Q_1, Q_2$  on kaks sellist mustrit ja  $Q_1 <_{pf} Q_2$ , siis  $P$  ühendamist mustriga  $Q_1$  proovitakse enne kui mustriga  $Q_2$ . Olgu  $R$  muster, mis sellise ühendamise tulemusel saadakse. Kui  $R$  on maksimaalne ja esineb vähemalt  $K$  erinevas sisendsekventsisis, siis pannakse  $R$  pinu tippu,  $R$  saab uueks käesolevaks tipuks ja protseduur algab jälle otsast. Vastasel korral  $R$  jäetakse kõrvale, käesolev tipp jääb muutmata ja proovitakse järgmist ühendamist.

Maksimaalsuse kontrollimiseks rakendatakse meetodit, mis seab igale mustrile  $R$  vastavusse teatava räsifunktsiooni väärtuse  $H(R)$  ning otsib maksimaalsete mustrite hulgast  $M$  vastava väärtusega paisktabeli kirjet. Funktsioon  $H$  valitakse selline, et

1. kahel erineval maksimaalsel mustril ei oleks ühesugust väärtust ja
2. iga maksimaalsele mustrile  $R$  vastava mittemaksimaalse mustri  $R'$  korral  $H(R) = H(R')$

Nii avastatakse mittemaksimaalne muster vähima ressursikuluga, sest muster  $R$  on maksimaalne ainult siis, kui tema räsile vastav kirje paisktabelis on tühi.

Kui muster  $P$  pinu tipus ei ole enam sufiksi suunas (paremale) laiendatav, rakendatakse sama protsessi prefiksi suunas. Selleks otsitakse hulgast  $DirP$  mustrit  $Q$ , mille korral  $P \oplus Q \neq \emptyset$ . Jälle, kui leidub enam kui üks selline muster  $Q$ , siis kasutatakse osalist järjestust  $<_{sf}$ , et määrata kindlaks ühendamiste järjekorda.

Eelnevast järeldub intuiitiivselt, et kui laiendamine mõlemas suunas on lõppenud, on pinu tipus maksimaalne muster. Viimane eemaldatakse pinust, pannakse maksimaalsete mustrite hulka  $M$  ja raporteeritakse. Protsess algab uuesti pinu järgmise tipuga. Tsükel lõpeb, kui pinus ei ole järel ühtegi mustrit.

Kirjeldatud protseduuri korratakse, kuni elementaarmustrite hulk  $S_E$  on tühi.

## 4 Algoritmi rakendused

Meetodi väljatöötajad, IBM'i bioinformaatika ja mustrite avastamise grupp (*IBM's Bioinformatics and Pattern Discovery group*) on oma töö katsetamiseks ka Internetis kättesaadavaks teinud. Kirjeldatud algoritmi erinevaid rakendusi võib leida aadressil <http://cbcsrv.watson.ibm.com/Tspd.html>, samas on üleval ka kõikide programmide koodid ning abimaterjalid, samuti teised nimetatud uurimisgrupi projektid [2].

Teiresias'e (*Sequence Pattern Discovery*) kasutajal on võimalik valida kahe variandi, täpse ja kattuva vahel. Teisel juhul peab kasutaja sisestama sümboligrupid, mis võivad sekventsides teineteist asendada. Enamlevinud grupid on valitavad rippmenüüst. Uuritava sekventsise hulga võib samuti valida kas rippmenüüst või käsitsi sisestada.

Algoritmi poolt genereeritud mustrid kuvatakse ekraanil sajakaupa, on olemas nende hulgas navigeerimise vahendid, samuti mitmed omaduste (nt. minimaalne/maksimaalne esinemise sagedus, esinemistõenäosus jne.) järgi mustrite filtreerimise võimalused. Lisaks saab konkreetse genereeritud mustri esinemisi otsida sisendsekventsides (mustri vastav sekvensi osa on paksus punases kirjas) või andmebaasist SWISS-PROT/TrEMBL (kuvatakse baasi kirjed, milles antud muster esineb).

Antud leheküljel on võimalik tutvuda ka teiste Teiresias'el põhinevate rakendustega. Üks selline on mõeldud näiteks positiivsete täisarvude hulgast huvitavate korrapärasuste otsimiseks (*Integer-based Pattern Discovery*, <http://cbcsrv.watson.ibm.com/Tipd.html>). Seda omakorda kasutatakse tavateksti töötlemiseks mõeldud rakenduses, milles käsitletakse iga sisendis olevat sõna eraldi üksusena (*Text-word Pattern Discovery*, [/Twpd.html](#)). Ning lõpuks veel üks tavateksti töötlev rakendus, milles vaadeldakse sisendit sümbolhaaval (*Text-symbol Pattern Discovery*, [/Ttspd.html](#)).

## 5 Kokkuvõte

Käesolevas töös vaatlesime põhjalikumalt mustrite otsimise algoritmi Teiresias. Nimetatud algoritm vaatab ainult potentsiaalseid lahendeid (st piisava esinemissagedusega maksimaalseid mustreid), nõuetele mittevastavad mustrid jäetakse vaatluse alt välja võimalikult vara. Seetõttu sõltub programmi töötamise aeg ainult väljastatavate mustrite arvust, mis väga sarnaste sisendsekventsise korral võib osutada äärmiselt suureks ja sellest tulenevalt algoritmi töötamise aeg eksponentsiaalseks. Tegelikuses aga esineb selliseid olukordi harva ja algoritmi soorituse aeg jääb suhteliselt mõistlikuks ka suurte sisendhulkade korral. Uurisime põgusalt ka veebikeskkonda, kus saab kasu-

tada nimetatud algoritmi erinevaid rakendusi.

Mustrite otsimise temaatika ja meetodite kohta annab väga hea ülevaate artikkel [1], kus käsitletakse antud teemat bioloogilistest andmetest tähendust omavate seaduspärade otsimise valguses. Samuti on selles artiklis toodud mitmete rakenduste aadressid Internetis.

## References

- [1] B. Brejova, C. DiMarco, and T. Vinar. Finding Patterns in Biological Sequences. Technical report, University of Waterloo, 2000.
- [2] T. Huynh, I. Rigoutsos, L. Parida, D. Platt, and T. Shibuya. The web server of IBM's Bioinformatics and Pattern Discovery Group. *Nucleic Acids Research*, 31(13):3645–3650, 2003.
- [3] B. Liu, W. Hsu, L.-F. Mun, and H.-Y. Lee. Finding interesting patterns using user expectations. Technical report, National University of Singapore.
- [4] V. Mäkinen, G. Navarro, and E. Ukkonen. Approximate Matching of Run-Length Compressed Strings, 2002.
- [5] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. *Bioinformatics*, 14(1):55–67, 1998.
- [6] I. Rigoutsos and A. Floratos. On the Time Complexity of the TEIRESIAS Algorithm. Technical report, IBM T. J. Watson Research Center, 1998.
- [7] A. Wespi, H. Debar, and M. Dacier. An Intrusion-Detection System Based on the Teiresias Pattern-Discovery Algorithm. Technical Report RZ3103, Zurich Research Laboratory, IBM Research Division, 1999.