UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA MATEMATIKO

Živa Stepančič

# UPORABA STATISTIČNIH METOD PRI RAZISKOVANJU MOTIVOV V DNA

Doktorska disertacija


MENTOR: doc. dr. Damjana Kokol Bukovšek
SOMENTOR: prof. dr. Matjaž Omladič

Ljubljana, 2015

Podpisana Živa Stepančič izjavljam:

- da sem doktorsko disertacijo z naslovom *Uporaba statističnih metod pri raziskovanju motivov v DNA* izdelala samostojno pod mentorstvom doc. dr. Damjane Kokol Bukovšek in somentorstvom prof. dr. Matjaža Omladiča in
- da Fakulteti za matematiko in fiziko Univerze v Ljubljani dovoljujem objavo elektronske oblike svojega dela na spletnih straneh.

Ljubljana, 8.6.2015 Podpis:

# ZAHVALA

Za uspešno izvedbo doktorskega raziskovanja se zahvaljujem svoji mentorici doc. dr. Damjani Kokol Bukovšek in somentorju prof. dr. Matjažu Omladiču. Najlepša hvala tudi dr. Gregorju Šegi za njegove nasvete pri razvijanju nove metode za iskanje motivov v DNA sekvencah. Zahvala gre še podjetju Arctur d.o.o. za vso tehnično podporo. Doktorsko delo je nastalo ob finančni podpori javne agencije Republike Slovenije SPIRIT in EU.

Posebna zahvala gre mojim staršem, ki so mi s svojimi nasveti pomagali pri raziskovanju in mi dali potrebno moralno podporo, da sem svoje delo pripeljala do zaključene celote podane v tej disertaciji. Iskrena hvala tudi moji najboljši prijateljici Antoniji, ki je z razumevanjem in spodbudo izjemno olajšala pot do zaključka doktorskega raziskovanja.

# UPORABA STATISTIČNIH METOD PRI RAZISKOVANJU MOTIVOV V DNA


## Povzetek

V tem delu predstavimo novo metodo za iskanje motivov v DNA sekvencah, ki izhaja iz Gibbsovega vzorčenja, kot ga je na to področje uvedel Charles Lawrence s sodelavci v članku [27]. Za izboljšavo osnovnega postopka Gibbsovega vzorčenja smo razvili model, s katerim določimo začetek iterativnega postopka in povečamo verjetnost konvergence Gibbsovega vzorčevalnika. Metoda je v osnovi razdeljena na dva dela: prvi del naše metode zajema modeliranje z grafom, v drugem delu pa uporabimo Gibbsov vzorčevalnik. Algoritem smo poimenovali *GraphGibbs*.

Raziskovalni problem je iskanje *neznanih* motivov v DNA sekvencah. Motiv predstavlja vzorec oziroma zaporedje nukleotidov, ki se v množici sekvenc pojavi statistično nadpovprečno. V tem delu smo se osredotočili na iskanje vzorcev v segmentih DNA sekvence, na katere se vežejo tako imenovani transkripcijski dejavniki, ki uravnavajo prepis dedne informacije z molekule DNA na molekulo RNA za potrebe delovanja organizma. Ti vzorci pa nimajo nujno identičnega zaporedja nukleotidov, kar pomeni, da je motiv samo relativno ohranjen. V tem primeru ima motiv pozitivno stopnjo variabilnosti, ki jo določajo porazdelitve DNA nukleotidov na posameznem mestu zaporedja po vseh vzorcih, ki imajo isto funkcionalnost.

Od odkritja dvojne vijačnice in genskega koda v 70. letih 20. stoletja se je raziskovalni fokus na področju genetike preusmeril v razumevanje uravnavanja genskega izražanja. Gensko izražanje v veliki meri regulirajo transkripcijski dejavniki, zato obstaja veliko zanimanje za določanje segmentov v DNA sekvenci, na katere se ti dejavniki vežejo. Za iskanje veznih mest oziroma motivov so se razvile različne metode, ki jih v grobem ločimo na metode preštevanja, verjetnostne metode in metode strojnega učenja. Gibbsovo vzorčenje je verjetnostna metoda, ki išče rešitev lokalno in uporabi Bayesov princip določanja porazdelitve iskanih parametrov v modelu.

Neznani parametri tovrstnih metod so položaji pojavitev motiva v sekvencah. Ti položaji so zbrani v urejen seznam, ki mu pravimo poravnava. Cilj algoritma je torej najti poravnavo motiva. Ker so motivi neznani, kar pomeni, da ne poznamo niti njegovega položaja v sekvenci niti zaporedja nukleotidov v zapisu, lahko statistično opredelimo problem iskanja motivov v DNA sekvencah kot problem manjkajočih podatkov. Dober pristop za reševanje tega problema je tudi Gibbsovo vzorčenje, kjer iterativno določimo ciljno porazdelitev neznanih in neopaznih parametrov modela.

Za boljšo natančnost in hitrejšo konvergenco k ciljni porazdelitvi nepoznanih parametrov smo k osnovnemu algoritmu dodali predhodno obdelavo DNA sekvenc, pri kateri kodiramo sekvence in jih modeliramo z multigrafom. Za kodiranje DNA sekvenc smo uporabili kar genski kod in tako razširili abecedo sekvenc iz štirih črk na 64 črk, kolikor je tudi trojčkov po genskem kodu. Frekvence pojavitev para sosednih trojčkov v celi množici sestavljajo matriko povezav, iz katere dobimo matriko sosednosti za utežen multigraf na 64 vozliščih. Glede na porazdelitev uteži določimo

induciran podgraf na desetih vozliščih z najbolj uteženimi izhodnimi povezavami. Analiza tako dobljenega induciranega podgrafa nam omogoča določitev popolnoma ohranjenega vzorca s statistično nadpovprečnim številom pojavitev v celi množici.

S poravnavo popolnoma ohranjenega vzorca dobimo dodatne informacije o notranji sestavi danih DNA sekvenc. Z dodatno informacijo poiščemo porazdelitev števila pojavitev motiva po sekvencah kot tudi delno določimo začetno točko pri Gibbsovem vzorčevalniku. Oboje izboljša natančnost Gibbsovega vzorčenja in pospeši konvergenco h končni točki. Točka pri Gibbsovem vzorčenju predstavlja poravnavo motiva in jo na vsakem koraku ocenimo s statistiko $I$, ki predstavlja "informacijo na parameter". Rešitev algoritma je poravnava, ki ima največjo vrednost statistike $I$.

Za uporabo algoritma *GraphGibbs* potrebujemo samo množico analognih segmentov DNA sekvenc, v kateri želimo poiskati motiv in njegovo poravnavo. Možne so različne uporabniške nastavitve, med drugim nastavitev števila različnih motivov, dolžine motivov in števila pojavitev motiva v dani množici. Vpliv prostih dejavnikov na rezultat algoritma, ki smo ga merili s statistiko $I$, smo preverili s Plackett-Burman eksperimentalnim načrtom na generiranih testnih množicah. Posebej smo generirali še množice za testiranje upešnosti algoritma *GraphGibbs*. Dodatno smo preverili delovanje naše metode na realnih podatkih.

Uspešnost algoritma *GraphGibbs* merimo z njegovo občutljivostjo in pozitivno napovedno vrednostjo kot tudi s tako imenovanim koeficientom uspešnosti in korelacijskim koeficientom med znano in najdeno poravnavo motiva. Dodatno smo izračunali še specifičnost algoritma *GraphGibbs* na različnih generiranih in realnih množicah. Na generiranih množicah smo raziskali osnovne karakteristike motiva (dolžina motiva) in njegove poravnave (število pojavitev in njihova porazdelitev) glede na število in dolžino vhodnih sekvenc, pri katerih je algoritem *GraphGibbs* najbolj uspešen. Pri realnih množicah smo naredili primerjavo uspešnosti našega algoritma na sekvencah štirih različnih organizmov, in sicer človeka, miši, muhe in glive. Primerjalno bazo realnih množic so sestavili Martin Tompa in sodelavci, s katero so statistično analizirali uspešnost trinajstih različnih metod za iskanje motivov in svoje izsledke predstavili v članku [48]. Na izbranem vzorcu realnih množic se je izkazalo, da je algoritem *GraphGibbs* zelo uspešen na sekvencah človeka, miši in muhe v primerjavi z ostalimi algoritmi. Na celotnem primerjalnem vzorcu realnih množic se je algoritem *GraphGibbs* pokazal kot peti najuspešnejši algoritem med štirinajstimi algoritmi, priloga B.

Rezultati algoritma *GraphGibbs* na različnih generiranih množicah so pokazali, da algoritem dobro najde relativno ohranjene motive, predvsem kadar je njihov zapis daljši od devet nukleotidov in je približno deset sekvenc v vhodni množici. Prav tako lahko uspešno loči med različnimi motivi, kar smo preverili na generiranih množicah z dvema motivoma. Pri testiranju uspešnosti naše metode na realnih množicah smo primerjali različne nastavitve algoritma. Največjo občutljivost algoritma na realnih množicah smo zaznali pri nastavitvi, kjer smo omejili samo število možnih pojavitev motiva.

Z razširjeno raziskavo vpliva različnih lastnosti motiva, kot sta njegova dolžina, število pojavitev in porazdelitev števila pojavitev, se lahko naš model za iskanje motivov še izboljša. Prav tako so potrebne dodatne obdelave DNA sekvenc več različnih organizmov, da lahko podamo bolj natančno oceno uspešnosti našega algoritma. Metodo je mogoče še nadgraditi, tako da omilimo vpliv nekaterih predpostavk, ki smo jih uporabili pri raziskovanju rešitve za problem iskanja motivov v DNA sekvencah.

**Math. Subj. Class. (2010):** 62K15, 62P10, 65C05, 68R10, 90C35
**Ključne besede:** iskanje motivov, DNA sekvence, graf, Gibbsovo vzorčenje

## STATISTICAL METHODS FOR MOTIF FINDING IN DNA

Abstract

In this work we present a new method for finding motifs in DNA sequences. It is based on Gibbs sampling, introduced to this field by Charles Lawrence with colleagues, [27]. To improve the basic method using Gibbs sampling, we develop a model, where we partially determine the beginning of the iterative process, and improve the probability of Gibbs sampler converging. Our method is basically divided into two parts: the first part uses graphical modelling of initial DNA sequences, while the second part consist of Gibbs sampler. We have named the algorithm *GraphGibbs*.

Problem of finding *unknown* motifs in DNA sequences is the central part of our research. Motif is represented by a pattern, or rather a sequence of nucleotides, which are statistically overrepresented in the given set of sequences . In our work we focus on finding motifs in DNA sequences that correspond to binding sites of so called transcription factors. The transcription factor regulates the transmission of gene information from DNA molecule to RNA molecule in protein building process. The corresponding patterns are not necessarily identical as they can be relatively conserved. Relative conservation implies positive degree of variability, which is determined with individual distributions of DNA nucleotides at all positions of every motif instance in the set.

Since the discovery of double helix structure of a DNA molecule and genetic code in the 1970s, the focus of genetics research turned to solving the regulation process of gene expression. Gene expression is regulated primarily with transcription factors, thus there is great interest in identifying segments in DNA sequences that contain factors' binding sites. There are many motif finding methods, which can be divided into word counting methods, probabilistic methods and machine learning methods. Gibbs sampling is a probabilistic method that searches for solution locally, and uses the Bayesian principle for determining the distribution of model parameters.

Unknown model parameters in motif finding methods are the binding sites of a motif in the set. The binding sites are grouped into an ordered list that represents motif's alignment. The goal of a motif finding algorithm is thus to find the alignment of a motif. Since the motifs are unknown, that is their alignments as well as sequence of nucleotides are unknown, we can define the motif finding problem in DNA sequences as a missing data problem. A good approach towards solving this problem is also Gibbs sampling method, where the target distribution of unknown and unobserved parameters are determined through an iterative process.

To improve accuracy and increase the rate of convergence towards target distribution of unknown parameters, we added a pre-processing step to Gibbs sampling algorithm, with which coded DNA sequences are modelled with a multigraph. Sequences are coded with genetic code, which consists of 64 triplets. We define a connection matrix using frequency of sequential pairs of triplets occurring in the whole set, from which we construct an incidence matrix for a weighted multigraph of order 64. We

select ten vertices whose outgoing arcs have the highest weights, and analyse the subgraph induced by these vertices. By analysing the structure of induced graph via walks of certain length, we can determine completely conserved patterns that are statistically overrepresented inside the whole set.

The alignment of completely conserved patterns gives extra information about the structure of DNA sequences given. This extra information helps us determine the distribution of motif sites in the set, as well as partially define the starting point of the Gibbs sampler. Both considerably improve the accuracy and convergence of the Gibbs sampling process. In each iteration of Gibbs sampler we gain a motif alignment, which is evaluated by statistic $I$. The letter $I$ stands for an abbreviation for "information per parameter". The motif alignment with highest value of statistic $I$ is the solution.

Input data essential to *GraphGibbs* algorithm is a set of DNA sequences. Additionally, a user can make assumptions on the number of motifs in the set, their lengths and the number of expected sites. The effect of individual free user parameter on the value the algorithms' solution, which is measured by the statistic $I$, was evaluated in a Plackett-Burman experimental design on generated test data sets. We tested the performance of *GraphGibbs* algorithm on another group of generated data sets in addition to real data sets.

To measure algorithms' performance we used the following statistics: sensitivity, positive predictive value, specificity and performance coefficient of the algorithm, together with correlation coefficient between the known and predicted alignment. On generated data sets we analysed possible combinations of different motif characteristics (motif length, number and distribution of motif sites) and data set features (number and length of sequences in the data set) to find a combination best suited for *GraphGibbs* algorithm. To analyse the performance on real data sets of human, mouse, fly and yeast sequences, we used a benchmark data base compiled by Martin Tompa and his colleagues. With this data base they compared performance of thirteen motif finding algorithms via statistical analysis, and posted their findings in their article [48]. *GraphGibbs* algorithm shows very good overall performance on human, mouse and fly sequences in comparison to other thirteen algorithms. Comparison on a chosen sample placed *GraphGibbs* algorithm as fifth best among all fourteen algorithms according to overall performance, appendix B.

Results of *GraphGibbs* algorithm on various generated data sets imply good performance on sets with approximately ten sequences and with motif length of nine nucleotides or longer, even at higher degrees of motif variation. On generated data sets with two known motifs we confirmed a successful separation of different motifs for most combinations of user parameters. We compared effects of different combinations of user parameters on real data sets as well. Algorithm is most sensitive to all real data sets when we only assign a value for expected number of motif sites among all possible user parameters.

To improve our method for motif finding, an expended study is necessary to see

how various motif characteristics, such as motif length, expected number of motif sites and their distribution in the set, effect the performance of the algorithm. Furthermore, an additional analysis on multiple real data sets of DNA sequences from different species is integral for building a more successful model for applicable purposes. On the other hand, an upgrade can be achieved by mitigating or even removing some of the basic assumptions made in the course of our investigation of motif finding methods and in development of its solution.

**Math. Subj. Class. (2010):** 62K15, 62P10, 65C05, 68R10, 90C35
**Keywords:** motif finding, DNA sequences, graph, Gibbs sampling

# Kazalo

# Seznam slik

# Seznam tabel

# Seznam algoritmov

# 1. Uvod

Eden največjih izzivov v biologiji je razbrati in razumeti mehanizme, ki regulirajo zapise v *deoksiribonukleinski kislini*(DNA). Pri tem je pomembno identificirati regulatorne elemente v genski sekvenci, še posebej vezna mesta encimov, ki kontrolirajo prepisovanje dedne informacije na *informacijsko ribonukleinsko kislino* (mRNA). Vezna mesta so kratki segmenti v DNA, ki jih imenujemo *motivi*. Iskanje motivov je kompleksen problem na več področjih, med drugim genetike, molekularne biologije, medicine, računalništva, matematičnega modeliranja in statistike.

Odkrivanje postopka dedovanja se je začelo z raziskavami Gregorja J. Mendela, ki je leta 1866 postavil preprosto teorijo o dedovanju lastnosti iz generacije v generacijo. Svojo teorijo je osnoval na poskusih križanj rastlin, ki so se razlikovale po določenih vidnih lastnostih, kot so velikost rastline, barva cvetov ter oblika in barva semen. Mendelova teorija je prišla v ospredje šele leta 1900, ko so bila objavljena dela treh raziskovalcev, to so Hugo de Vries, Carl Correns in Erich von Tschemerk. Petnajst let kasneje se je Mendelova teorija povezala z Boveri-Suttonovo teorijo o kromosomskem dedovanju in postavila osnove *klasične genetike* [50].

Klasična genetika se ukvarja predvsem z načini dedovanja lastnosti, opisovanjem dednih lastnosti in proučevanjem kromosomov. Zaključki so dobljeni na vidnih posledicah pri reprodukciji organizmov. Z raziskovanjem dednih lastnosti in njihovim prenosom iz generacije v generacijo znotraj organizma se ukvarja posebna veja genetike, ki jo imenujemo *molekularna genetika*. Raziskovanje poteka z biokemijskimi metodami na molekularnem nivoju, kjer se proučuje zgradba in funkcija odsekov v molekuli DNA, ki nosi dedno informacijo. Na podlagi genskih analiz znanstveniki klasificirajo oziroma razvrščajo organizme v sistematske enote, ki se ločijo glede sorodstvenih odnosov med vrstami. Takšno sistematično razvrščanje organizmov imenujemo *filogenija* [49].

Molekularna genetika se je osnovala v drugi polovici 20. stoletja. Prej so mnogi znanstveniki menili, da je informacija zapisana v beljakovinah [46]. Razkritje tridimenzionalne zgradbe DNA je ta prepričanja ovrglo in usmerilo znanstveno raziskovanje v verigo DNA in njeno funkcijo za življenje. Eno izmed najbolj udarnih odkritij je bilo odkritje dvojne vijačnice DNA. Model molekule DNA, kot sta ga postavila James D. Watson in Francis H.C. Crick [52], je postala prava ikona 20. stoletja. V svojem članku sta predstavila svoj model molekularne strukture nukleinskih kislin, ki ga je znanost privzela kot pravilnega. Odkritje dvojne vijačnice je bilo

omogočeno z rezultati rentgenske kristalografije, s katero se je ukvarjala kemičarka Rosalind Franklin pod vodstvom Mauricea Wilkinsa [46].

Osredja dogma molekularne genetike na kratko orišemo s shemo

$$\text{DNA} \Longleftrightarrow \text{RNA} \Longrightarrow \text{protein} \tag{1.1}$$

Povezave v shemi 1.1 povedo, da za sintezo beljakovin potrebujemo načrt (gen) v molekuli DNA, ki se nahaja v jedru celice in molekulo RNA (mRNA) za prenos načrta iz jedra v telo celice, kjer se nahajajo proste aminokisline za izgradnjo proteinov. Najprej so menili, da ta pot poteka samo v eno smer, vendar se je pri raziskovanju genomov nekaterih virusov (RNA virusov, retrovirusov) pokazalo, da se po infekciji v gostiteljevi celici zapis iz RNA virusa obratno prepiše v molekulo DNA. Od tod naprej pa teče informacija samo v eno smer.

Beljakovine so nujno potrebne za življenje, zato je pomembno razumeti celoten proces njihove izgradnje pri živih bitjih. Proces se začne z aktiviranjem posebnih encimov, ki se vežejo na odgovarjajoče dele v molekuli DNA in vzpostavijo začetek prepisovanja dedne informacije z verige DNA na verigo mRNA. Te pomembne odseke DNA imenujemo tudi motivi. Za odkrivanje lokacij motivov in njihove funkcionalnosti pri izgradnji proteinov v genskih sekvencah potrebujemo:

- verjetnostno strukturo za generiranje genske verige,
- strategijo za določitev dolžine,
- učinkovito strategijo za optimizacijo računske zahtevnosti algoritma,
- verjetnostni model, ki dopušča napake v zapisu motiva kot je menjava črk v motivu ali delitev zapisa motiva.

Identifikacija motivov je zahtevna naloga, saj so motivi ponavadi zelo majhen del genske verige (so dolžine $6-21$ baznih parov), ki jih iščemo v množicah s sekvencami dolžine $100-3000$ baznih parov. Takšno razmerje povzroči veliko statističnega šuma pri določanju motivov. Razvoj metod za iskanje aktivnih motivov dodatno otežuje še variabilnost zapisa motivov in delitev motivov na dva dela z različno dolgim razmikom med deloma.

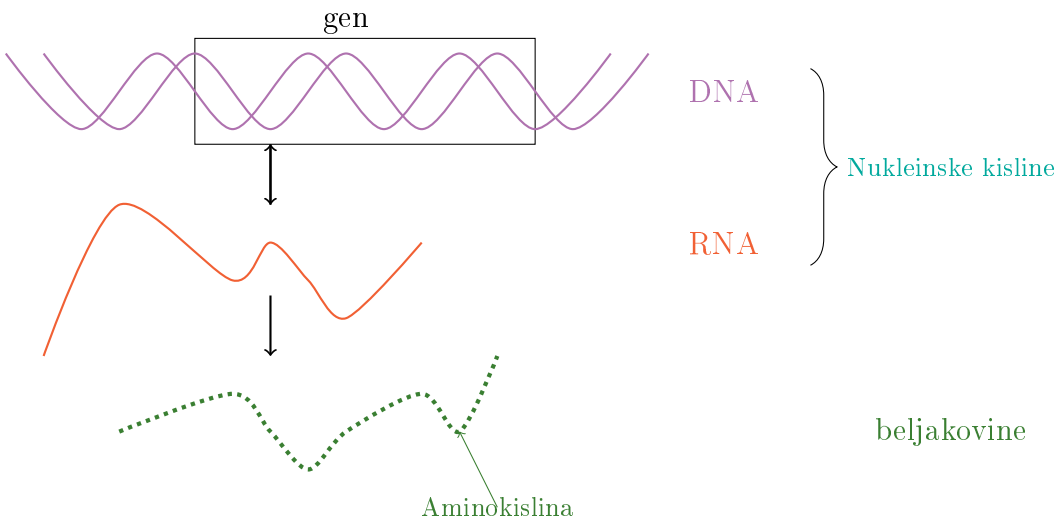
Metode in algoritmi, ki se ukvarjajo z reševanjem problema iskanja motivov se zanašajo predvsem na statistično razmišljanje. Razvoj statističnih teorij podpira in pospešuje razvoj algoritmov v bioinformatiki kot tudi na drugih področjih. Z raznimi hevrističnimi prijemi so ocene težjih teoretičnih enačb in izračunov vedno bolj primerne za nadaljno analizo. Rezultate lahko pridobimo hitreje kot z analitičnim postopkom, kar je v praksi lahko velika prednost. Dober približek odnosov in razmerij med posameznimi parametri znotraj raziskovalnega sistema omogoča boljšo diagnozo problema in hitrejše reševanje. Slednje je v kontekstu časovno občutljive situacije še posebej pomembno.

## 1.1 Biološko ozadje

Dešifriranje zapisa v molekuli DNA skupaj s postopkom njenega prenosa iz molekule DNA je trenutno v žarišču več disciplin, predvsem na področjih molekularne genetike, medicine, genskega inženirstva in bioinformatike. Kot nosilka dedne informacije je molekula DNA temelj za nadaljevanje življenja in je eden izmed devetih osrednjih bioloških konceptov [46]. Dedna informacija je shranjena v krajših odsekih v molekuli DNA, ki jih imenujemo *geni*. V vsakem genu je zapisan načrt za izdelavo beljakovin, ki jih organizem potrebuje.

Poleg klasifikacije zapisa genov in njihove funkcionalnosti potekajo še raziskave o *uravnavanju genskega izražanja*. Gensko izražanje je izraz za prenos genskega zapisa v protein. Za nastanek proteina je potrebno precej korakov. Poenostavljena shema je podana v sliki 1.1, ki oriše osredji koncept v molekularni genetiki 1.1. Genska informacija se z DNA molekule prenese z ribonukleinsko kislino (ang. ribonucleic acid), na kateri poteka izgradnja pomembnih aminokislin [46].

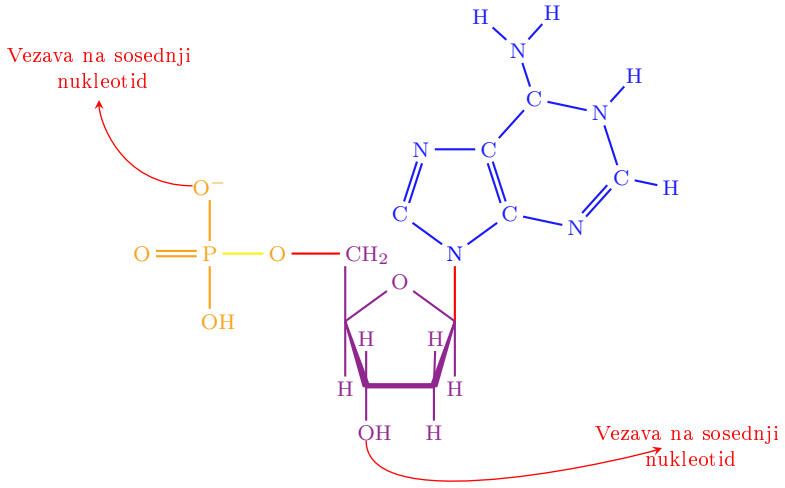
Molekula DNA je makromolekula kakor mnoge druge organske molekule v celici. Makromolekula je velika molekula, ki jo celica izgradi s povezovanjem sto ali tisoč manjših molekul. Ti osnovni gradniki se imenujejo monomeri, ki se sestavljajo v daljše verige polimerov. Glavni polimeri za delovanje organizma so: beljakovine, ki so zgrajene iz aminokislin; škrob, zgrajen iz molekul glukoze; in nukleinske kisline, ki so zgrajene iz nukleotidov. Obstaja še ena pomembna skupina organskih snovi, lipidi, med katerimi ni makromolekul.




Slika 1.1: Shema prenosa dedne informacije do izgradnje beljakovine.

Čeprav nosilko dedne informacije enačimo z molekulo DNA, je to le en tip nukleinskih kislin, ki se nahajajo v vsaki celici. Drugi tip so molekule ribonukleinske kisline ali RNA. Oba tipa sta polimera zgrajena iz nukleotidov, ki so monomeri, sestavljeni iz atomov ogljika, kisika, dušika, vodika in fosforja. Poznamo tri skupine, ki gradijo molekulo nukleotida, in sicer sladkorno bazo, fosfatno skupino in dušikovo bazo.

V molekuli DNA se monomeri imenujejo DNA-nukleotidi. Na sliki 1.2 je shema zgradbe DNA-nukleotida. Prikazana je kemijska struktura treh delov nukleotida, pri čemer sta fosfatna skupina in sladkorna baza enaki pri vseh DNA-nukleotidih, medtem ko so dušikove baze različne [46]. Na sliki je za prikaz vzeta dušikova baza adenin.

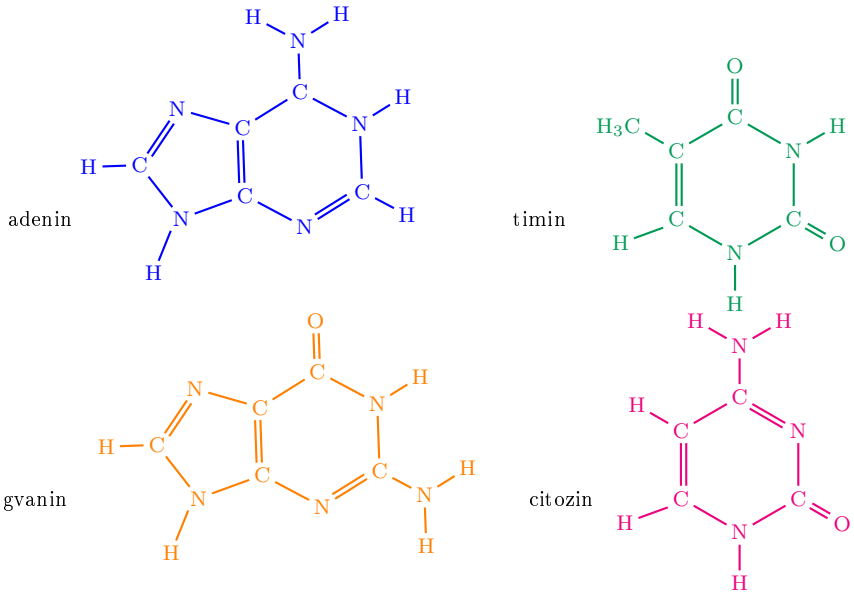



Slika 1.2: Shema nukleinske kisline sestavljene iz treh delov: fosfatne skupine, sladkorja (deoksiriboze) in dušikove baze (adenin).

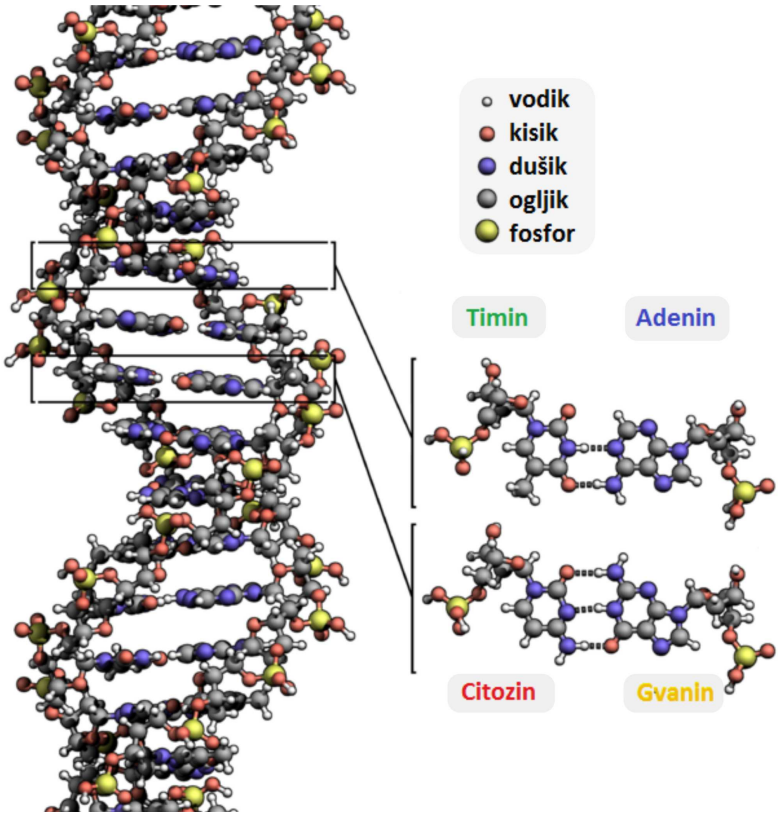
Sladkorno bazo, ki se imenuje *deoksiriboza*, predstavlja v obroč postavljenih pet ogljikovih atomov. Vsak ogljikov atom se poveže s štirimi drugimi atomi, ki so lahko del molekule ali pa samostojni. Na deoksiribozo je na eni strani vezana negativno nabita fosfatna skupina, na drugi pa dušikova baza. Na enem od atomov je vezana še hidroksilna skupina (-OH). Preostale vezi pa zasedajo samostojni atomi vodika in kisika.

V DNA-nukleotidu poznamo štiri različne dušikove baze, ki jih lahko vidimo na sliki 1.3, in sicer *adenin* (A), *timin* (T), *citozin* (C) in *gvanin* (G). S kraticami A,T,C in G označimo kar cel nukleotid, saj se le-ti razlikujejo samo po dušikovih bazah. Dušikove baze se med seboj vežejo z vodikovimi vezmi in tvorijo bazne pare. Molekulo DNA tako sestavlja zaporedje baznih parov, kar ilustrira tudi slika 1.4. Fosfatna skupina se na deoksiribozo veže z estrsko vezjo. Preko te skupine se tako povezujejo sladkorne baze nukleotidov. Ker obstajata dve estrski vezi za dve deoksiribozi, sta dva nukleotida povezana s tako imenovano fosfordiestrsko vezjo. Na ta način se posamezni nukleotidi povezujejo v verigo [12]. Dve verigi nukleotidov se zvijeta druga okrog druge v tako imenovano dvojno vijačnico.

Model dvojne vijačnice DNA, kot se je uveljavil v znanstvenih krogih, sta postavila James D. Watson in Francis H.C. Crick [52]. Ugotovila sta, da se med dušikovimi bazami posameznih nukleotidov izoblikujejo posebne vodikove vezi, ki lahko nastanejo samo med dušikovimi bazami. Dve vodikovi vezi sta med adeninom in timinom ter tri vodikove vezi med citozinom in gvaninom. Dvojna vijačnica in vodikove vezi, ki gradijo bazna para, so prikazane v spodnji ilustraciji 1.4.




Slika 1.3: Dušikove baze v nukleinski kislini, s katerimi se tvorita dva bazna para v molekuli DNA, in sicer adenin - timin in gvanin - citozin.

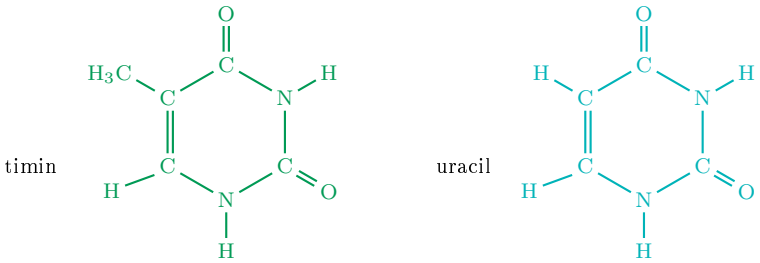



Slika 1.4: Shema dvojne vijačnice DNA molekule z atomi in z vodikovimi vezmi, ki gradijo bazna para, [54].

Verigi v vijačnici sta vzporedni, a usmerjeni v nasprotni smeri ali antiparalelni. Še več, verigi sta komplementarni, saj se med seboj povezujejo komplementarne baze in ne identične baze. Na podlagi te ugotovitve sta Watson in Crick pravilno predvidevala, da se dvojna veriga odpre kot zadrga med baznimi pari, tako da se porušijo vodikove vezi med njimi. Te se nato nazaj vzpostavijo, ko se veriga "zapre". Model Watsona in Cricka pojasni, kako je molekula zgrajena in kako so geni lahko različno dolga zaporedja nukleotidov.

Genska informacija v molekuli DNA je osrednja pri izgradnji proteinov, vendar obstaja še en pomemben dejavnik pri procesu nastanka beljakovin, in sicer molekula informacijske RNA oziroma mRNA (ang. messenger RNA), kot vidimo v shemi 1.1. Molekula RNA je manj izpostavljena, a je ključ celotnega procesa in s kronološkega vidika celo pomembnejša od DNA. Znanstveniki namreč predvidevajo, da so prva živa bitja nastala na osnovi RNA [51].

Molekule RNA se lahko vedejo kot molekule DNA v smislu shranjevanja in podvajanja genske informacije in hkrati se obnašajo kot beljakovina, ki katalizira pomembne kemične reakcije. Sestavljene so iz ribonukleotidov, ki so enaki DNA-nukleotidom z izjemo dušikove baze timin. Namesto timina je v RNA dušikova baza uracil, ki je komplementarna baza bazi adenin. Kemijska struktura uracila je prikazana na sliki 1.5 poleg strukture timina za primerjavo. V nasprotju z DNA so vse molekule RNA enoverižne.




Slika 1.5: Dušikova baza timin na levi in uracil na desni strani.

V molekuli RNA so dušikove baze v ribonukleotidih tako komplementarne dušikovim bazam v DNA-nukleotidih. Ta lastnost omogoča proces *prepisovanja gena* ali transkripcije, kjer se zaporedje nukleotidov v genu prepiše na RNA verigo. Na začetek gena v vijačnici DNA se veže poseben encim, imenovan polimeraza RNA. Verigi DNA se razkleneta in polimeraza RNA začne polagati na sproščene DNA-nukleotide kodirajoče verige komplementarne ribonukleotide in jih poveže med seboj v verigo RNA. Polimeraza RNA se pomika po genu, dokler ne pride do konca gena, kjer naleti na zaključno zaporedje, ki ga prepiše v konec RNA verige. Povezave med komplementarnimi bazami RNA verige povzročijo, da se veriga RNA zvija v prostorsko obliko, ki je pomembna za njeno delovanje. Po zaključku transkripcije se polimeraza RNA odcepi od kodirajoče verige, ki se nazaj poveže z drugo verigo DNA v obliki vijačnice [21].

S procesom transkripcije se začne gen "izražati". Izraz *izražanje gena* pomeni gra-

dnjo RNA molekule in pri večini genov še odgovarjajoči protein, ki ima neko funkcijo v celici ali pa izven nje. Poleg polimeraze RNA pa so potrebni še posebni sprožilni dejavniki oziroma specifične proteinske molekule, ki se s svojo obliko prilagajajo odgovarjajoči obliki *regulizacijskega zaporedja* gena, kjer izpostavijo začetek gena encimu polimeraze RNA.

Regulacijsko zaporedje pred posameznim genom, na katerega se vežejo regulatorne beljakovine, imenujemo *promotor* (ang. promoter region). Pri prokariotskih celicah so geni razporejeni tako, da je potreben samo en promotor, ki nadzira cel niz zaporednih genov, ki so potrebni za izgradnjo beljakovine. Promotor skupaj z določenim nizom genov sestavljaja transkripcijsko enoto, ki jo imenujemo *operon*. Nadzorno območje operona poleg promotorja sestavlja še operator. Nanj se vežejo posebne beljakovine, ki jih imenujemo zaviralci ali represorji, ki polimerazi RNA preprečujejo začetek transkripcije. Represorje izdeluje poseben, ločen regulacijski gen, ki ima lasten, neprestano delujoč promotor. Kot pove ime, ti encimi zavirajo začetek transkripcije, zato se bo encim polimeraze RNA vezal na DNA verigo šele, ko se bo represor sprostil z območja operatorja. Za to poskrbi posebna molekula, ki jo imenujemo vzpodbujevalec ali induktor. Takšen način uravnavanja izražanja genov imenujemo negativno uravnavanje aktivnosti.

V nasprotju s prokariontskimi celicami pri ekvariontskih celicah geni niso povezani v operone, saj ima večina genov lastne promotorje. Prav tako pri transkripciji pogosteje kot represorji sodelujejo aktivatorji. Aktivatorji so beljakovine, ki vključijo gensko izražanje. Pri evkariontski transkripciji aktivatorje imenujemo *transkripcijski dejavniki* [36] (ang. transcription factors), ki se vežejo na promotorje. Z vezavo aktivirajo vezavo encima polimeraze RNA na promotor in s tem povzročijo začetek transkripcije. Kadar uravnavanje izražanja genov izvajajo aktivatorji, potem govorimo o pozitivnem uravnavanju aktivnosti. Pri takšnem delovanju so pri mnogoceličnih evkariontskih organizmih geni večino časa "izklopljeni" oziroma se ne prepisujejo na molekulo RNA [12]. Transkripcija se aktivira, ko jih celica potrebuje. Stalno so vklopljeni le geni, ki skrbijo za rutinske operacije celice, kot je na primer celično dihanje.

Regulacija genov je v osnovi regulacija procesa transkripcije, to je prepisa gena iz DNA sekvence na RNA sekvence. Za transkripcijo potrebujemo dovolj odprto kromatinsko strukturo, do katere imajo dostop regulatorne molekule. Za dejanski proces pa so potrebni encimi RNA polimeraze, ki kopirajo zapis iz DNA sekvence na RNA sekvenco skupaj z več vrstami transkripcijskih dejavnikov, ki aktivirajo ali zavirajo polimerizacijo.

Pri prokariontih je dovolj en encim RNA polimeraze, ki je odgovoren za kopiranje zapisa z DNA na RNA. Drugače je pri evkariontih, kjer je takšnih encimov več in jih ločimo na tri skupine; imenujemo jih RNA polimeraze I, II in III. V rastlinah obstajata še dva dodatna tipa, in sicer RNA polimeraza IV in V. Encimi RNA polimeraze I, II in III so veliki in sestavljeni iz več enot. Nekatere enote lahko najdemo pri encimih v vseh treh skupinah. Vendar se encimi med seboj deterministično razlikujejo glede na relativno občutljivost na glivični strup $\alpha$-amanitin in vsaka skupina

je aktivna na specifični skupini genov [26].

Za izgradnjo proteinov je pomemben encim RNA polimeraza II, kjer je transkripcija genov tudi bolj kompleksna, v nasprotju z encimoma RNA polimeraze I in III. Transkripcija genov pa je samo en proces pri izgradnji proteinov. Pomemben je še proces *translacije*, kjer se zaporedje ribonukleotidov na verigi mRNA "prevede" v zaporedje aminokislin. Translacija se izvaja na posebnih enotah v celicah, imenovanih ribosomi.

Veriga mRNA je kombinacija štirih različnih ribonukleotidov. V naravi najdemo dvajset različnih aminokislin, ki z raznimi kombinacijami gradijo beljakovine. Pri postopku translacije mora *ribosom* po nekem ključu šifrirano zaporedje ribonukleotidov prevesti v želeno zaporedje aminokislin. Ključ je v 60. letih 19. stoletja odkril Crick s sodelavci, slabih sedem let po njegovem prispevku pri modelu dvojne vijačnice. V njihovem udarnem članku [8] so avtorji predstavili *genski kod* na podlagi eksperimentalnih poskusov. Kod je sistem znakov za spreminjanje informacij iz ene oblike v drugo. Na ravni molekule DNA se dedna informacija iz zaporedja štirih različnih DNA-nukleotidov prenese v obliki skupka treh zaporednih nukleotidov, ki se prepiše na molekulo RNA in ga imenujemo *kodon*. Dvajset aminokislin in odgovarjajoči kodoni so predstavljeni v tabeli 1.1.

Tabela 1.1: Genski kod. V RNA je timin (T) zamenjan z uracilom (U). Trije kodoni ne kodirajo nobene aminokisline, ampak označujejo zaključek translacije.

| **Aminokislina** | **Kodon RNA** | | | | | |
|---|---|---|---|---|---|---|
| alanin | GCA | GCC | GCG | GCU | | |
| arginin | AGA | AGG | CGA | CGC | CGG | CGU |
| asparagin | AAC | AAU | | | | |
| asparaginska kislina | GAC | GAU | | | | |
| cistein | UGC | UGU | | | | |
| glutaminska kislina | GAA | GAG | | | | |
| glutamin | CAA | CAG | | | | |
| glicin | GGA | GGC | GGG | GGU | | |
| histidin | CAC | CAU | | | | |
| izolevcin | AUA | AUC | AUU | | | |
| levcin | UUA | UUG | CUA | CUC | CUG | CUU |
| lizin | AAA | AAG | | | | |
| metionin | AUG | | | | | |
| fenilalanin | UUC | UUU | | | | |
| prolin | CCA | CCC | CCG | CCU | | |
| serin | AGC | AGU | UCA | UCC | UCG | UCU |
| treonin | ACA | ACC | ACG | ACU | | |
| triptofan | UGG | | | | | |
| tirozin | UAC | UAU | | | | |
| valin | GUA | GUC | GUG | GUU | | |
| *zaključni kodon* | UAA | UAG | UGA | | | |

Po članku [8] ima genski kod naslednje lastnosti:

- Skupina treh nukleinskih baz ali tudi *trojček* (ang. triplet) kodira eno aminokislino.
- Kod se ne prekriva.
- Zaporedje nukleinskih baz se bere s fiksne začetne točke. V primeru, da je začetna točka zamaknjena, se zamakne tudi branje trojčkov in posledično dobimo napačno zaporedje aminokislin.
- Kod je izrojen ali degeneriran. To pomeni, da določenim aminokislinam odgovarja več različnih trojčkov.

Zadnja lastnost je bila še pred objavo članka izpostavljena teoretično kot posledica možnega kombiniranja štirih nukleotidov za določanje 20 aminokislin. V primeru, da bi aminokislino kodiral kar nukleotid, bi lahko določili samo štiri aminokisline. Tudi s pari nukleotidov ne bi mogli dobiti vseh aminokislin, kajti število vseh kombinacij dveh baz je samo 16. Torej so potrebni vsaj trije nukleotidi, da lahko zapišemo vseh 20 aminokislin. Vendar pri kodiranju s trojčki dobimo 64 možnih kombinacij, kar je pa presežek za določitev 20 aminokislin. Torej je ista aminokislina lahko zapisana z več kot eno kodno besedo (kodon), kar je prikazano v tabeli 1.1.

## 1.2 Opis problema

Najprej opredelimo nekaj osnovnih pojmov, ki jih potrebujemo za formulacijo našega raziskovalnega problema.

Dano imamo končno množico znakov $\mathcal{A} = \{b_1, b_2, \ldots, b_K\}$, $K \in \mathbb{N}$, ki ji pravimo *abeceda*. Znaki (ang. token) so lahko simboli ali črke. Prav tako imamo dan niz $R$ končne dolžine $L \in \mathbb{N}$, ki je sestavljen z znaki iz $\mathcal{A}$. Niz $R$ lahko obravnavamo kot *besedilo*, ki ga želimo analizirati. Za poljubno naravno število $w < \infty$ lahko tvorimo *besede* dolžine $w$ kot končne nize sestavljene iz znakov abecede $\mathcal{A}$. Zanima nas, katere besede dolžine $w$ se najpogosteje pojavijo v danem besedilu $R$.

**Definicija 1.2.1.** *Večkratna množica* je množica z več enakimi elementi.

Označimo večkratno množico $M_w$ kot množico vseh besed dolžine $w < L$, ki se pojavijo v besedilu $R$. Posamezne besede se lahko v množici večkrat pojavijo, saj je $M_w$ večkratna množica. Število kopij poljubne besede $v$ dolžine $w$ v množici $M_w$ označimo kot $num[M_w](v)$. Tako je moč množice $M_w$ :

$$|M_w| = \sum_{v \in M_w} num[M_w](v).$$

Z elementi množice $M_w$ lahko oblikujemo podmnožice, ki vsebujejo besede enakega pomena ali sopomenke. Sopomenke so besede, ki opisujejo enako funkcionalnost, vendar se lahko razlikujejo med seboj v zaporedju znakov. Vse podmnožice zberemo v množico, ki jo označimo z $\mathcal{P}M_w$.

**Definicija 1.2.2.** *Vzorčna množica* je množica besed določene dolžine enakega pomena. Elementom vzorčne množice pravimo *vzorčni nizi*.

V splošnem, kadar analiziramo besedila, imamo določena pričakovanja o številu pojavitev nekaterih besed znotraj besedila. Za zgled lahko vzamemo matematično besedilo o grafih napisano v slovenskem jeziku, kjer lahko pričakujemo pogostejše pojavitve besed 'graf', 'povezava', 'vozlišče'. Odgovarjajoče sopomenke bi lahko bile 'omrežje', 'lok', 'krajišče'. Podobna imamo pričakovanja glede števila pojavitev določenih besed dolžine $w$ znotraj besedila $R$. Število pričakovanih pojavitev označimo z $E \in \mathbb{N}$. Želimo najti vzorčne množice znotraj množice $\mathcal{P}M_w$, katerih moč je večja ali enaka številu $E$.

**Definicija 1.2.3.** *Presečni vzorec* $v$ je predstavnik vzorčne množice $A$, vendar ni nujno tudi njen element in ni enolično določen. Označimo ga z $v_A$.

Presečni vzorec $v_A$ vzorčne množice $A \subseteq \mathcal{P}M_w$ določimo iz njenih vzorčnih nizov. Za vsako mesto $i = 1, \ldots, w$ presečnega vzorca določimo znak $b_j$, $j = 1, \ldots, K$, glede na porazdelitev frekvenc znakov na mestu $i$ pri vseh vzorčnih nizih. Pogledamo torej, kateri znak $b_j$ ima na določenem mestu $i$ pri vsakem od vzorčnih nizov najvišjo frekvenco $c_{ij}$ in ga postavimo na mesto $i$ presečnega vzorca. Na primer, če velja

$$c_{ir} = \max_{j=1,\ldots,K} c_{ij}, \quad i = 1, \ldots, w,$$

potem je na mestu $i$ presečnega vzorca $v$ znak $b_r$, kar označimo kot $v^i = b_r$.

Lahko se zgodi, da ima več znakov na določenem mestu enako (najvišjo) frekvenco in dobimo več kandidatov za presečni vzorec. Med kandidati izberemo enega slučajno, kar pomeni z enako verjetnostjo, kot predstavnika vzorčne množice. Tako zgrajen presečni vzorec ni nujno tudi element vzorčne množice, za katero je predstavnik.

**Definicija 1.2.4.** Za vzorčno množico $A$ s presečnim vzorcem $v_A$ je *stopnja variabilnosti* $d(v_A)$ največje število mest v posameznem vzorčnem nizu, kjer je znak različen od znaka na istoležnem mestu v presečnem vzorcu.

Nakazali smo že, da obstaja vzajemna povezava med vzorčno množico $A$ ter stopnjo variabilnosti presečnega vzorca $d(v_A)$. Vzorčni nizi so zbrani v množice glede na njihov pomen oziroma na določeno lastnost, a se lahko med seboj razlikujejo. Število različnih znakov na enakoležnih mestih vzorčnih nizov in presečnega vzorca $v_A$ določa stopnjo variabilnosti $d(v_A)$. Prav tako pa lahko sestavimo vzorčno množico s presečnim vzorcem in z vnaprej določeno stopnjo variabilnosti. Stopnja variabilnosti med drugim opredeli tudi *ohranjenost* presečnega vzorca $v_A$.

**Definicija 1.2.5.** Presečni vzorec $v_A$ je popolnoma ohranjen , kadar so vzorčni nizi enaki in je stopnja variabilnosti $d(v_A) = 0$.

**Definicija 1.2.6.** Presečni vzorec $v_A$ dolžine $w$ je relativno ohranjen , kadar je stopnja variabilnosti $d(v_A) \leq w/4$.

Osnovni raziskovalni problem je iskanje *neznanih* relativno ohranjenih zaporedij elementov oziroma podnizov v danem nizu. Kadar se vzorčni nizi med seboj preveč razlikujejo ali drugače, kadar je stopnja variabilnosti vzorčnih nizov prevelika, potem presečni vzorec ni ohranjen in ne odraža vzorčne množice. Takšne množice nas ne zanimajo. Zanimajo nas množice, ki imajo vsaj relativno ohranjene presečne vzorce, saj vsaj ena izmed njih predstavlja rešitev problema.

**Opomba 1.2.7.** Kadar za dve vzročni množici $A$ in $B$ velja $v_A = v_B$, potem trdimo, da sta vzorčni množici enakovredni.

**Definicija 1.2.8.** *Vzorec* je presečni vzorec vzorčne množice, ki jo vzamemo kot rešitev.

V najbolj enostavnem modelu je niz $R$ zaporedje elementov iz $\mathcal{A}$ končne dolžine $L$. Vzorec je podniz dolžine $W$, $W \in \mathbb{N}$, ki se v nizu $R$ pojavi vsaj dvakrat in je popolnoma ohranjen. Cilj raziskovalca je določiti mesta, kjer se vzorec pojavi in določiti njegovo zaporedje.

**Primer 1.2.9.** Predstavimo zgoraj definirane pojme na enostavnem abstraktnem problemu s simboli. Naj bo dana abeceda $\mathcal{A} = \{\times, \triangle, *\}$ s tremi različnimi simboli. Naj bo besedilo dano kot niz simbolov iz $\mathcal{A}$ dolžine $L = 29$ :

$$R = \times \times * * * \triangle * * \times \triangle \triangle \times \times \times * \times \triangle \triangle \times * * * \times \triangle \triangle \triangle * \triangle \triangle \triangle .$$

Dano imamo še dolžino besed $w = 4$ in število pričakovanih pojavitev iskanega vzorca $E = 3$. Naš cilj je poiskati vzorčno množico moči 3, katere presečni vzorec je relativno dobro ohranjen. Vzorčna množica, ki je sprejemljiva, mora imeti stopnjo variabilnosti $\leq 1$, saj je pri dolžini $w = 4$ sprejemljiv le en različen znak v zaporedju vzorčnega niza od zaporedja presečnega vzorca oziroma kar vzorca.

Sestavimo večkratno množico $M_4$ vseh besed dolžine 4, katere moč je

$$|M_4| = L - (w - 1) = 26.$$

$$\begin{aligned} M_4 = \{ & \times \times * *, \times * * *, * * * \triangle, * * \triangle *, * \triangle * *, \triangle * * \times, * * \times \triangle, * \times \triangle \triangle, \\ & \times \triangle \triangle \times, \triangle \triangle \times\times, \triangle \times \times \times, \times \times \times *, \times \times * \times, \times * \times \triangle, * \times \triangle \triangle, \\ & \times \triangle \triangle \times, \triangle \triangle \times *, \triangle \times * *, \times * * \times, * * \times \triangle, *\times \triangle \triangle, \times \triangle \triangle \triangle, \triangle \triangle \triangle *, \\ & \triangle \triangle * \triangle, \triangle * \triangle \triangle, * \triangle \triangle \triangle\} \end{aligned}$$

Zanimajo nas vzorčne množice moči 3. Vzorčne množice sestavljajo množico $\mathcal{P}M_4$, katere moč je enaka

$$\binom{|M_4| + E - 1}{E} = \binom{26 + 3 - 1}{3} = \binom{28}{3} = 3276.$$

Torej imamo 3276 možnih rešitev, vendar so spremenljive samo vzorčne množice s stopnjo variabilnosti $\leq 1$. Izpišimo nekaj vzorčnih množic:

$$A_1 = \{\times\ \times\ *\ *, *\ \triangle\ \triangle\ \triangle, \times\ \times\ *\ \times\}$$
$$A_2 = \{*\ *\ *\ \triangle, \triangle\ *\ *\ \times, \times\ *\ *\ \times\}$$
$$A_3 = \{\times\ \triangle\ \triangle\ \times, \times\ \triangle\ \triangle\ \times, \times\ \triangle\ \triangle\ \triangle\}$$
$$A_4 = \{\times\ \times\ *\ *, \times\ \times\ *\ \times, \times\ *\ *\ \times\}$$
$$\vdots$$

Zapišimo jih v obliko tabele in določimo vse presečne vzorce. V tabelo 1.2 smo zapisali vzorčne nize štirih vzorčnih množic $A_1, A_2, A_3$ in $A_4$. Vzorčni nizi so zloženi navpično, da lahko lažje določimo kandidate za presečni vzorec kar s prostim očesom. Na posamezno mesto presečnega vzorca postavimo simbol, ki se pojavi najpogosteje na istoležnem mestu v vzorčnih nizih.

Tabela 1.2: Primeri štirih vzorčnih množic iz množice $\mathcal{P}M_4$ s kandidati za presečni vzorec.

| **Vzorčna množica** | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| **Vzorčni nizi** | $\times\ \times\ *\ *$<br>$*\ \triangle\ \triangle\ \triangle$<br>$\times\ \times\ *\ \times$ | $*\ *\ *\ \triangle$<br>$\triangle\ *\ *\ \times$<br>$\times\ *\ *\ \times$ | $\times\ \triangle\ \triangle\ \times$<br>$\times\ \triangle\ \triangle\ \times$<br>$\times\ \triangle\ \triangle\ \triangle$ | $\times\ \times\ *\ *$<br>$\times\ \times\ *\ \times$<br>$\times\ *\ *\ \times$ |
| **Kandidati za presečni vzorec** | $\times\ \times\ *\ *$<br>$\times\ \times\ *\ \triangle$<br>$\times\ \times\ *\ \times$ | $*\ *\ *\ \times$<br>$\triangle\ *\ *\ \times$<br>$\times\ *\ *\ \times$ | $\times\ \triangle\ \triangle\ \times$ | $\times\ \times\ *\ \times$ |
| **Presečni vzorec** | $\times\ \times\ *\ *$ | $\times\ *\ *\ \times$ | $\times\ \triangle\ \triangle\ \times$ | $\times\ \times\ *\ \times$ |
| **Stopnja variabilnosti** | 4 | 2 | 1 | 1 |

Sestavimo presečni vzorec za množico $A_1$. Spomnimo se, da presečni vzorec ni enolično določen niti ni nujno pripadnik vzorčne množice. Porazdelitev frekvenc simbolov na mestu $i = 1, 2, 3, 4$ označimo kot $c_i = \{c_{i\times}, c_{i\triangle}, c_{i*}\}$, kjer kot prej $c_{ij}$ predstavlja frekvenco pojavitev znaka $b_j$, $j \in \{\times, \triangle, *\}$, na mestu $i$ preko vseh vzorčnih nizov v $A_1$. Tako dobimo naslednje porazdelitve za vsa štiri mesta:

$$c_1 = \{2, 0, 1\}$$
$$c_2 = \{2, 1, 0\}$$
$$c_3 = \{0, 1, 2\}$$
$$c_4 = \{1, 1, 1\}$$

Na posamezno mesto v presečnem vzorcu vstavimo simbol z najvišjo frekvenco, to je za vsak $i$ izberemo znak $b_j$, za katerega velja $v^i_{A_1} = \max_j c_{ij}$. Pri množici $A_1$ dobimo tako tri kandidate za presečni vzorec, kajti frekvenčna porazdelitev za zadnje, četrto, mesto je enakomerna. Opazimo tudi, da eden izmed kandidatov ni pripadnik množice $A_1$. Predstavnika množice med kandidati izberemo slučajno z verjetnostjo $p = 1/3$, saj so vsi enako verjetni.

Ostale presečne vzorce določimo analogno. Pri množicah $A_3$ in $A_4$ sta presečna vzorca določena enolično in pripadata svojima vzorčnima množicama. Stopnja variabilnosti $d_{A_3} = d_{A_4} \leq 1$, kar pomeni, da je presečni vzorec relativno ohranjen in tako sprejemljiv kot rešitev problema. Tako dobimo več možnih rešitev, ki jih lahko ovrednotimo glede na funkcionalnost vzorca. Slednje je del interpretacije rešitve, ki je, tako kot v tem delu, v večini primerov zunaj konteksta matematičnega modeliranja danega problema.

♦

Pri naši raziskavi je bil problem iskanja neznanih vzorcev postavljen v genetski okvir. Tukaj niz $R$ predstavlja skupek sekvenc DNA istega tipa organizma, ki so lahko različnih dolžin. Abeceda $\mathcal{A}$ je sestavljena iz štirih nukleinskih baz. V tem kontekstu raziskovalci vzorcem rečejo tudi motivi.

**Definicija 1.2.10.** *Motiv* je iskan, neznan, relativno ohranjen vzorec neznane dolžine, ki ima v dani množici sekvenc DNA neko funkcijsko nalogo za delovanje organizma.

V nadaljevanju bomo uporabljali izraz motiv izključno kot vzorec definiran v definiciji 1.2.8. Cilj je najti motiv v skupku sekvenc DNA, določiti njegovo vzorčno množico in položaje vzorčnih nizov. Bolj formalno to pomeni, da želimo najti motiv, skupen danim sekvencam DNA in njegovo poravnavo.

**Definicija 1.2.11.** Naj bo $S = \{S_1, \ldots, S_N\}$ množica $N$ DNA sekvenc in $\boldsymbol{P} = (p_1, \ldots, p_N)$ vektor števila pojavitev motiva v vsaki od sekvenc iz $S$. *Poravnava* $A(v)$ je seznam $N$ vektorjev, kjer $k$-ti vektor $\boldsymbol{a_k} = (s_1, \ldots, s_{p_k})$ predstavlja indekse pojavitev vzorca $v$ v sekvenci $S_k$.

**Primer 1.2.12.** Opišimo problem iskanja motivov v DNA sekvencah na enostavnem primeru, kjer se popolnoma ohranjen motiv $v$ dolžine $w = 6$ pojavi enkrat na sekvenco. Pri DNA sekvencah je abeceda $\mathcal{A}$ sestavljena iz štirih nukleinskih baz, to je $\mathcal{A} = \{\text{A,T,C,G}\}$ in $K = 4$. Podano imamo tudi množico $S = \{S_1, \ldots, S_N\}$ z $N = 3$ DNA sekvencami iste dolžine $\ell = 15$. Združimo jih skupaj v eno dolgo sekvenco $R = S_1 + S_2 + S_3$ dolžine $L = 45$. Operator $+$ predstavlja spojitev dveh

sekvenc $S_i$ in $S_j$, tako da zadnji črki v $S_i$ sledi prva črka sekvence $S_j$.

$S_1 = \text{AACGACTCTTTAAGA}$
$S_2 = \text{CCTATCATTTAAGGA}$
$S_3 = \text{TAACCTTTTAAGGGA}$

$R = \text{AACGACTCTTTAAGACCTATCATTTAAGGATAACCTTTTAAGGGA}$

Ker iščemo en motiv na sekvenco, je pričakovano število pojavitev motiva v sekvenci $R$ enako $E = 3$. Motiv $v$ sestavlja zaporedje šestih črk, zato si poglejmo vse besede dolžine $w = 6$, ki jih lahko preberemo v treh sekvencah. Moč množice $M_6$ je potem enaka $|M_6| = (\ell - (w-1)) \cdot N = 30$.

$$\begin{aligned} M_6 = \{&\text{AACGAC, ACGACT, CGACTC, GACTCT, ACTCTT, CTCTTT,} \\ &\text{TCTTTA, CTTTAA, TTTAAG, TTAAGA, CCTATC, CTATCA,} \\ &\text{TATCAT, ATCATT, TCATTT, CATTTA, ATTTAA, TTTAAG,} \\ &\text{TTAAGG, TAAGGA, TAACCT, AACCTT, ACCTTT, CCTTTT,} \\ &\text{CTTTTA, TTTTAA, TTTAAG, TTAAGG, TAAGGG, AAGGGA}\} \end{aligned}$$

Vseh vzorčnih množic, ki sestavljajo $\mathcal{P}M_6$ je

$$\binom{|M_6| + E - 1}{E} = \binom{32}{3} = 4960.$$

Tako kot v primeru 1.2.9 si poglejmo nekaj primerov vzorčnih množic iz $\mathcal{P}M_6$, ki vsebujejo po tri vzorčne nize. Na primer:

$$\begin{aligned} A_1 &= \{\text{ACGACT, ATTTAA, CTTTAA}\} \\ A_2 &= \{\text{ACTCTT, ATCATT, CCTTTT}\} \\ A_3 &= \{\text{CTTTAA, CATTTA, TTTTAA}\} \\ A_4 &= \{\text{TTTAAG, TTTAAG, TTTAAG}\} \\ &\vdots \end{aligned}$$

Postavimo množice in vzorčne nize v tabelo 1.3 in poglejmo kandidate za presečne vzorce. Kjer je več kandidatov, potem vzamemo enega izmed njih slučajno kot presečni vzorec množice. Rešitev je presečni motiv vzorčne množice, ki ima stopnjo variabilnosti enako 0, saj iščemo popolnoma ohranjen motiv $v$.

Poglejmo si določitev presečnega vzorca in stopnje variabilnosti množice $A_1$. Postopek za ostale množice je enak. Najprej določimo kandidate za presečni vzorec, za kar poglejmo porazdelitev pojavitve posamezne črke na vsakem mestu $i = 1, \ldots, 6$. Označimo porazdelitev posameznih črk na mestu $i$ kot $c_i = \{c_{i\text{A}}, c_{i\text{T}}, c_{i\text{C}}, c_{i\text{G}}\}$, kjer je $c_{ij}$ število pojavitve črke $j \in \{\text{A,T,C,G}\}$ na mestu $i$ pri vseh vzorčnih nizih. S

Tabela 1.3: Primeri štirih vzorčnih množic iz množice $\mathcal{P}M_6$ s kandidati za presečni vzorec.

| **Vzorčna množica** | $A_1$ | $A_2$ | $A_3$ | $A_4$ |
|---|---|---|---|---|
| **Vzorčni nizi** | ACGACT<br>ATTTAA<br>CTTTAA | ACTCTT<br>ATCATT<br>CCTTTT | CTTTAA<br>CATTTA<br>TTTTAA | TTTAAG<br>TTTAAG<br>TTTAAG |
| **Kandidati za presečni vzorec** | ATTTAA | ACTCTT<br>ACTATT<br>ACTTTT | CTTTAA | TTTAAG |
| **Presečni vzorec** | ATTTAA | ACTTTT | CTTTAA | TTTAAG |
| **Stopnja variabilnosti** | 5 | 3 | 2 | 0 |
| **Motiv** | | | | TTTAAG |

temi oznakami dobimo naslednje porazdelitve:

$$c_1 = \{2, 0, 1, 0\}$$
$$c_2 = \{0, 2, 1, 0\}$$
$$c_3 = \{0, 2, 0, 1\}$$
$$c_4 = \{1, 2, 0, 0\}$$
$$c_5 = \{2, 0, 1, 0\}$$
$$c_6 = \{2, 1, 0, 0\}$$

Za presečni vzorec vzamemo največji element v $c_i$ za vsako mesto $i$. S takšno izbiro dobimo kandidata za presečni vzorec ATTTAA. Ker je to edini kandidat, je presečni vzorec kar $v_{A_1}$ = ATTTAA. Poglejmo še, kakšna je stopnja variabilnosti, ki jo dobimo s primerjavo med $v_{A_1}$ in vsakim od vzorčnim nizov v $A_1$. Naj $d_n$ predstavlja število mest, kjer se $v_{A_1}$ in vzorčni niz $n$ razlikujeta. Potem velja $d = \{d_1, d_2, d_3\} = \{5, 0, 1\}$. Stopnja variabilnosti $d(v_{A_1}) = \max_{i=1,2,3} d_i = 5$.

Pri ostalih množicah se presečni motivi in stopnje variabilnosti določi enako. Vzorčne množice, ki predstavljajo rešitev, so samo množice s stopnjo variabilnosti 0, kajti želimo najti popolnoma ohranjen vzorec. Za motiv vzamemo popolnoma ohranjen presečni vzorec množice $A_4$. To je tudi edini popolnoma ohranjen motiv, ki se pričakovano pojavi trikrat v množici $S$.

Ko smo določili rešitev $v$ = TTTAAG, si poglejmo še poravnavo $A(v)$. Elementi

$a_k \in A(v)$ poravnave določajo indeks začetka zapisa motiva $v$ v sekvenci $S_k$. Tako je poravnava $A(v) = \{9, 8, 7\}$

♦

Primera 1.2.9 in 1.2.12 enostavno orišeta raziskovalni problem ter hkrati nakažeta možne ovire pri načrtovanju metode za iskanje neznanih vzorcev oziroma motivov v poljubni množici podatkov. Pri primeru 1.2.9 vidimo, da imamo lahko več možnih rešitev za končni iskani vzorec. Obe rešitvi sta enakovredni glede na postavljene omejitve - število pričakovanih mest in stopnjo variabilnosti. Vendar so lastnosti, ki jih vzorca predpisujeta, lahko različne in neenakovredne glede na interes raziskave.

Pri primeru 1.2.12 je sicer motiv enolično določen, a s spremembo omejitev pri iskanju lahko najdemo tudi druge rešitve. Vzemimo za zgled, da je pričakovano število pojavitev popolnoma ohranjenega motiva v celi množici sekvenc enako dve. Potem bi imeli dve možni rešitvi, in sicer motiva TTTAAG in TTAAGG. Ko omilimo še drugo omejitev in nas zanimajo relativno ohranjeni vzorci, dobimo še več možnih rešitev.

Iz zgornjih opažanj lahko razberemo, da s spreminjanjem omejitev dobimo več ali manj rešitev. Spomnimo se, da so motivi raziskovalcu neznani, torej raziskovalec ne pozna njihove dolžine, števila pojavitev ali ohranjenosti vzorca. Pomanjkanje teh ključnih lastnosti motiva je velika ovira pri postavljanju robustnega modela in razvoju metode, ki hitro najde možne in relevantne rešitve.

Pri bolj zapletenih primerih iskanja motivov v DNA sekvencah so množice večje ($N \geq 3$) in sekvence daljše ($\ell > 100$) in posledično so množice $\mathcal{P}M_w$ za neko dolžino $w \geq 6$ večje. Globalni pregled vseh vzorčnih množic pri vseh različnih možnih $w$ bi vzel veliko računskega časa. Nekateri algoritmi za iskanje motivov [9], naredijo globalni pregled, vendar jim uspešnost upade pri bolj razvitih organizmih (evkariontih), saj je tam stopnja variabilnosti višja. Za lokalno iskanje pa so potrebne določene omejitve, saj pregledamo manj vzorčnih množic. Empirične predpostavke, ki jih raziskovalci pogosto postavijo, so naslednje:

1. omejitev dolžine motiva $w$ na interval $[6, 20]$;
2. število pričakovanih pojavitev motiva $E$ v celi množici sekvenc je navzdol omejeno z številom sekvenc $N$, ki sestavljajo množico, torej $E \geq N$;
3. motiv $v$ mora biti vsaj relativno ohranjen, torej $d(v) \leq w/4$.

Tako v praksi model postane bolj kompleksen. Vzorci so redko popolnoma ohranjeni. Pri realnih podatkih je pogosto variabilnost vzorčnih nizov precej visoka. Poleg tega imamo vzorčne množice, v katerih vzorčni nizi niso istih dolžin. Tukaj ni samo problem določiti moč takšne množice, ampak tudi določiti njen presečni vzorec.

Dodaten zaplet predstavlja še možnost, da niz $R$ vsebuje več različnih tipov vzorcev. Med njimi so lahko tudi vzorci, ki nimajo relevantne funkcije glede na podatke, ki

jih preučujemo. Takšni vzorci predstavljajo tako imenovani beli šum v podatkih, vendar niso cilj raziskave. Kako določiti pomembne vzorčne množice in ločiti med različnimi tipi vzorcev, je problem tako pri modeliranju in obdelavi podatkov kakor tudi pri interpretiranju rezultatov.

Velikokrat so motivi relativno majhni in z večjo stopnjo variabilnosti. Določen motiv se lahko pojavi v promotorju več različnih genov ali pa večkrat znotraj gena [9]. Osnovna oblika je pogosto povezana tudi s strukturno obliko beljakovin. Poznamo pa še dodatne oblike motivov, tako imenovane *palindromske motive* (ang. palindromic motifs) in *diadne motive* (ang. spaced dyad/gapped motifs). Palindromski motiv je zaporedje nukleotidov, ki je enako inverzu komplementa, na primer:

zaporedje: GAATTC
komplement: CTTAAG
inverz komplementa: GAATTC

Diadni motivi so razdeljeni na dva kosa, med katerima je zaporedje nukleotidov, vmesnik (ang. spacer). Transkripcijski dejavnik, ki se veže na diadni motiv je tako dimer, kar pomeni, da je sestavljen iz dveh pod-enot, ki se vežeta na DNA. Velikost vmesnika je večinoma določena, vendar zaporedje ni popolno ohranjeno, kar predstavlja dodaten problem za razvoj metode za iskanje takšnih motivov.

Raziskovalni problem lahko enostavno povzamemo: V dani končni množici sekvenc najdi neznan, relativno ohranjen motiv, ki je statistično nadpovprečno zastopan v množicah. V tem delu se osredotočimo na iskanje motivov v DNA sekvencah, ki pripadajo organizmom iste vrste, kjer želimo najti neznane, statistično nadpovprečno zastopane in relativno ohranjene motive, ki se nahajajo v promotorskih regijah za prepisovalne dejavnike. Statistično lahko klasificiramo problem iskanja motivov kot problem manjkajočih podatkov. Manjkajoči podatek v tem primeru je poravnava vzorčnih nizov iskanega motiva [15], [9], [32]. Izraz problem se bo v nadaljevanju nanašal na zgornjo formulacijo, kjer analiziramo DNA sekvence istega organizma.

## 1.3 Metode za reševanje raziskovalnega problema

Vse od odkritja strukture DNA molekule in genskega koda se je raziskovanje genetikov in mikrobiologov usmerilo v pojasnjevanje mehanizmov, ki uravnavajo gensko izražanje. Kljub velikemu napredku na področju računalništva in bioinformatike so regulatorni mehanizmi za gensko izražanje še zmeraj v veliki meri neraziskani. Ti mehanizmi vsebujejo veliko korakov, po katerih celica na podlagi genske informacije zgradi potrebno beljakovino. Med drugim še ni popolnoma razvidno kdaj in kje se prepis gena začne. Poznamo že veliko transkripcijskih dejavnikov, ki začnejo postopek transkripcije, a še vedno ne poznamo vseh zaporedij nukleotidov oziroma motivov, na katere se transkripcijski dejavniki vežejo.

Prve metode so se osredotočile na iskanje motivov v promotorskih regijah koreguliranih (ang. co-regulated) genov v enem genomu. Gena sta koregulirana, kadar

spadata pod enak regulatorni mehanizem, ali drugače, imata enako uravnavanje izražanja. Razširjena predpostavka je, da imajo koregulirani geni tudi enak ali zelo podoben zapis, to je zaporedje nukleotidov na modekuli mRNA. Tako sta dva gena enako izražena (ang. co-expressed genes). Predpostavlja se tudi, da imajo geni z enako funkcionalnostjo podobno zaporedje nukleotidov. Aksiom funkcionalne genomike je, da imajo geni s podobnim zapisom na mRNA isti mehanizem uravnavanja izražanja gena. Ta hipoteza povezuje podatke iz mikromrež (ang. microarrays) z modelom regulatorne mreže za gensko izražanje, vendar še ni bila direktno testirana na večjem vzorcu [1].

Povezava med koregulacijo genov in njihovim zapisom je tudi prisotna pri modeliranju algoritmov za iskanje motivov. Iskanje je osredotočeno na iskanje relativno ohranjenih vzorcev, ki so statistično nadpovprečno zastopani v dani množici DNA sekvenc koreguliranih genov. Statistično nadpovprečno pomeni, da se motiv pojavi bolj pogosto, kot je pričakovano za število slučajnih pojavitev. Ker pa DNA sekvence pripadajo samo enemu genomu, so metode bolj ali manj uspešne na različnih genomih. Pri poskušanju poenotenja modelov za uspešno obdelavo pri različnih vrstah so se razvile metode, ki vključujejo tako imenovano *filogenijo* ali evolucijsko zgodovino vrst (ang. phylogenetic footprinting). Raznolikost med organizmi prihaja ravno iz raznovrstnosti njihovih dednih značilnosti. Sposobnost prilagoditve organizma na spremembe okolja se odraža v mutacijah in razvoju genske informacije. Evolucijski razvoj posameznih vrst organizmov temelji na posebnem mehanizmu, ki ga poznamo tudi kot *naravni izbor* [46].

Predpostavljamo, da so se v teku evolucije v selektivnem postopku določeni funkcionalni zapisi spreminjali med sorodnimi vrstami počasneje kot preostali deli v DNA sekvenci. Primer, ki podpira to predpostavko, je primerjava zaporedja baz istega gena pri miši in človeku [51]. Takšne podobnosti funkcionalnih zapisov sorodnih vrst spodbujajo razvoj metod, ki izkoristijo te predpostavke in razširijo uspešnost algoritma na več različnih vrst organizmov. Ena izmed ovir pri razvoju takšnih metod je dostop do ortoloških DNA sekvenc različnih vrst. Ortološke sekvence (ang. orthologous sequences) so homologne sekvence različnih vrst organizmov s skupnim prednikom.

Na podlagi DNA sekvenc, v katerih iščemo motive, ločimo algoritme za iskanje motivov v DNA sekvencah v tri skupine po [9], in sicer:

1. Algoritmi, ki analizirajo promotorje koreguliranih genov enega genoma;

2. Algoritmi, ki analizirajo promotorje ortoloških sekvenc enega gena različnih genomov;

3. Algoritmi, ki analizirajo promotorje obeh tipov sekvenc.

Pri našem raziskovanju smo se osredotočili na iskanje motivov transkripcijskih dejavnikov na promotorjih koreguliranih genov enega genoma. Odkrivanje motivov pri posameznih vrstah genomov je kljub veliki količini podatkov še vedno precej pomanjkljivo. Prav tako nismo imeli dostopa do ortoloških DNA sekvenc različnih

organizmov, ki bi jih lahko vključili v naš model. Čeprav obstajajo baze genov, kot so na primer TRANSFAC, OrthoMCL, TreeBASE, HUGO in NCBI-UniGene, je izbor ustreznih sekvenc, ki vsebujejo promotorje istih genov, kompleksna naloga in izven okvira našega raziskovanja, kjer je v ospredju postavitev matematičnega modela sekvenc, s katerim iščemo relativno ohranjene motive z uporabo statističnih metod. Prav tako obstaja v bioinformatiki pomanjkanje primerjalnih podatkovnih baz oziroma množic sekvenc, ki bi služile za testiranje uspešnosti algoritma.

Algoritme za iskanje motivov lahko dodatno klasificiramo glede na metodo, s katero analiziramo različne tipe DNA sekvenc. V grobem jih delimo na:

1. metode preštevanja nizov (ang. word-based methods via enumeration),
2. verjetnostne metode in
3. metode strojnega učenja.

Metode preštevanja nizov temeljijo na kombinatoričnem pristopu preštevanja in primerjanja frekvenc pojavitev določenih zaporedij v različnih kombinacijah. Pregled množice je globalen in zato računsko precej zahteven. Dodatno smo omejeni pri iskanju motivov z določeno stopnjo variabilnosti, pri katerih se uspešnost odkritja "pravih" precej zmanjša. Najnovejše metode vključujejo tudi takšne možnosti, kjer pa je rezultate potrebno dodatno obdelati z metodami razvrščanja v skupine (ang. cluster analysis).

Tabela 1.4: Algoritmi za iskanje motivov z verjetnostnim pristopom, ki so razširjeni na področju bioinformatike.

| ♯ | **Algoritem** | **Avtor** | **Glavna metoda** | **Verzija** |
|---|---|---|---|---|
| 1 | od Herzt et al. | Hertz et al. [17] | Požrešna metoda | Osnova |
| 2 | Consensus | Hertz & Stormo [18] | Utežena matrika | Razširitev ♯ 1 |
| 3 | EM | Lawrence & Reilly [29] | EM algoritem | Razširitev ♯ 1 |
| 4 | MEME | Bailey & Elkan [2] | EM algoritem | Razširitev ♯ 3 |
| 5 | Gibbs Sampler | Lawrence et al. [27] | Gibbsovo vzorčenje | Osnova |
| 6 | AlignACE | Roth et al. [40] | Gibbsovo vzorčenje | Razširitev ♯ 5 |
| 7 | MotifSampler | Thijs et al. [47] | Gibbsovo vzorčenje | Razširitev ♯ 5 |
| 8 | BioProspector | Liu et al. [34] | Gibbsovo vzorčenje | Razširitev ♯ 5 |
| 9 | GibbsST | Shida [41] | Simulirano temperiranje[1] | Razširitev ♯ 5 |

[1]MCMV metoda s področja termodinamike (ang. simulated tempering)

Metode strojnega učenja so za svoj razvoj potrebovale predvsem računsko zmogljivejše računalniške procesorje za praktično obdelavo večjih množic podatkov. Med drugimi so za iskanje motivov uporabili genetske algoritme in umetne nevronske mreže [9]. Uporabili so tudi metodo podpornih vektorjev, kjer uporabijo tako zapise v sekvencah, ki niso del promotorske regije (negativni primeri), kot zapise promotorskih regij (pozitivni primeri) [22]. Izziv tovrstnih metod je postaviti mejo, s katero razločimo med negativnimi in pozitivnimi primeri.

Metode z verjetnostnim pristopom iščejo rešitev lokalno, a se zadovoljivo spopadejo z večmodalnimi problemi in z integracijo v več dimenzijah. Vodilne metode uporabijo Gibbsovo vzorčenje (ang. Gibbs sampling) ali algoritem maksimizacije matematičnega upanja (ang. Expectation Maximization algorithm). Del algoritmov z verjetnostnim pristopom je opisan v tabeli 1.4, kjer smo dodatno označili, ali je bil algoritem prvi te vrste ali pa razširitev določenega algoritma. Algoritmi so v tabeli oštevilčeni v prvem stolpcu. Izbor algoritmov smo povzeli po [9], kjer so navedene algoritme izpostavili kot bolj udarne na področju bioinformatike. Poleg imena avtorja se nahajajo še referenčne številke iz seznama literature.

## 1.4 Pregled doktorskega dela

V zgornjih odstavkih je bil postavljen okvir raziskovalne problematike tega doktorskega dela skupaj s krajšo predstavitvijo raziskovalnega problema. V nadaljevanju obnovimo potek iskanja rešitve in predstavimo orodja, s katerimi smo sestavili končni postopek za iskanje motivov v sekvencah DNA. Naš algoritem smo poimenovali *GraphGibbs*. Ime je naravni povzetek dveh osnovnih matematičnih in statističnih modelov, na katerih algoritem temelji in jih preučimo v nadaljnjih poglavjih.

V drugem poglavju povzamemo teorijo, ki smo jo uporabili pri gradnji našega algoritma. Opisan je pregled raziskovalnega in statističnega razmišljanja, ki nas je vodil pri reševanju problema. Podrobneje je opisan postopek Gibbsovega vzorčenja. Prav tako izpostavimo nekaj definicij in lastnosti iz teorije grafov, s katerimi modeliramo dane sekvence in postavimo izhodišče za iskanje rešitve.

Vsa nadaljnja poglavja predstavljajo originalen znanstveni prispevek k disertaciji. V tretjem poglavju je opisan algoritem *GraphGibbs*, ki ga lahko v grobem razdelimo na dva dela: predpriprava sekvenc in Gibbsovo vzorčenje za iskanje rešitve. Naš algoritem temelji na algoritmu, ki so ga leta 1993 razvili Charles E. Lawrence in sodelavci, ki so prvi uvedli metodo Gibbsovega vzorčenja v kontekstu iskanja motivov v sekvencah DNA in proteinskih sekvencah [27]. Njihov algoritem smo priredili glede na našo pred-obdelavo sekvenc, ki temelji na grafičnem modelu danih sekvenc.

V četrtem poglavju predstavimo podatkovne množice, ki smo jih uporabili za analizo uspešnosti algoritma *GraphGibbs*. Predstavimo Plackett-Burmanov eksperimentalni načrt za pregled vpliva prostih parametrov algoritma. Opisane so statistike, s katerimi smo naredili analizo rezultatov na opisanih podatkih.

V petem poglavju grafično predstavimo rezultate Plackett-Burmanovega eksperimentalnega načrta. Uspešnost algoritma *GraphGibbs* na realnih in generiranih množicah DNA sekvenc grafično interpretiramo. Na kratko predstavimo izsledke diagnostike konvergence Gibbsovega vzorčevalnika.

V šestem poglavju povzamemo nekaj raziskovalnih opomb in podrobneje analiziramo rezultate, ki smo jih prikazali v petem poglavju. Izpostavimo nekaj osnovnih predpostavk, ki smo jih upoštevali pri razvijanju metode in naše razloge za njihovo vključitev.

V sedmem poglavju je predstavljen zaključek tega doktorskega dela. V njem povzamemo temeljne rezultate naše raziskave v kontekstu problematike iskanja motivov v sekvencah DNA. Posebej povzamemo omejitve, s katerimi smo se srečali pri razvijanju metode za iskanje motivov v DNA sekvencah. Nakazane so tudi možne poti nadaljnjega raziskovanja izboljšav k algoritmu *GraphGibbs*.

# 2. Teoretične osnove

V tem poglavju na kratko predstavimo in razložimo postopke, metode in orodja, ki smo jih uporabili v toku raziskovanja. Najprej opišemo proces znanstvenega in statističnega raziskovanja, ki smo ga uporabili v tem delu. Nato sledi opis Bayesovega pristopa k statističnemu raziskovanju v okviru bioinformatike. Nadaljujemo s predstavitvijo Gibbsovega vzorčevalnika, ki smo ga uporabili kot osnovo naše rešitve. Poglavje zaključimo z obnovo osnovnih pojmov iz teorije grafov, ki so potrebni pri razvoju algoritma *GraphGibbs*.

## 2.1 Znanstveno in statistično raziskovanje

Cilj znanstvenega raziskovanja je odkrivanje in določanje odnosov med spremenljivkami, s katerimi lahko opišemo in definiramo raziskovalni problem ali vprašanje. Spremenljivke so lahko objekti ali ljudje v raziskovalnem problemu, kakor tudi določene karakteristike, povezane z njimi. Običajno imamo pri iskanju odgovora na znanstveno vprašanje vsaj eno izhodno spremenljivko, označeno kot $Y$ in eksperimentalne spremenljivke, označene kot $\mathbf{X} = \{X_i, i = 1, 2, \ldots, n\}$, kjer $n$ predstavlja število vplivov na interesno spremenljivko $Y$.

Ker raziskujemo obnašanje spremenljivke $Y$ glede na določene zunanje dejavnike, jo imenujemo tudi odvisna spremenljivka; vplivi $\mathbf{X}$ pa se imenujejo neodvisne spremenljivke. Raziskovalce predvsem zanima, ali je spremenljivka $Y$ povezana s posameznimi spremenljivkami $X_i$ in kakšne vrste je ta odnos. Kadar povezava obstaja, potem pravimo, da je povezava *pozitivna*, kadar z naraščanjem vrednosti $X$ narašča tudi vrednost $Y$; drugače povedano obstaja *pozitivna korelacija*. Korelacija je lahko tudi *negativna*, kadar z naraščanjem vrednosti ene spremenljivke pada vrednost druge spremenljivke.

Pravimo, da med spremenljivkama obstaja *vzročna povezava*, kadar so spremembe spremenljivke $X$ vzrok sprememb spremenljivke $Y$. Ta povezava je pogosto *neposredna*, torej tudi merljiva. Kadar obstaja še ena, *skrita* ali *latentna* spremenljivka $Z$, ki vpliva vzročno na obe spremenljivki $X$ in $Y$, in ta vpliv vzpostavi povezavo, govorimo o *posredni* povezavi med spremenljivkama $X$ in $Y$. Spremenljivke $Z$ ne moremo direktno meriti, a vidimo njen vpliv na $X$ in $Y$ iz opazovanj. Možno je tudi, da obstaja direktna povezava med spremenljivkama $X$ in $Y$, kakor tudi skrita spremenljivka $Z$, ki vpliva na njuno obnašanje. Bistvo znanstvenega raziskovanja je torej določiti, kako so spremenljivke med seboj povezane in kako močna je ta

povezava.

Osnovna načela znanstvenega pristopa so se razvila v obdobju renesanse. Takrat se je iz filozofskih okvirjev prehajalo k testiranju teorij na realnih podatkih. Osnovna načela povzemimo v štirih točkah [5]:

- Znanstvena hipoteza ne more biti nikoli absolutno resnična.
- Obstajati mora možnost, da ovržemo znanstveno hipotezo.
- Model je koristen, dokler ne dokažemo, da ne drži.
- Vedno postavi najpreprostejšo hipotezo, za katero ni dokaza, da je napačna - Ockhamova britev (ang. Ockham's razor).

Z upoštevanjem zgornjih načel znanstvenega raziskovanja postavimo matematične modele, ki opisujejo odnose med neodvisnimi spremenljivkami in odvisno spremenljivko. Neodvisnim spremenljivkam v našem modelu rečemo tudi eksperimentalne spremenljivke, saj s spreminjanjem njihovih vrednosti merimo spremembe v spremenljivki, ki nas zanima. Za natančnejše rezultate moramo vse spremenljivke izolirati, tako da zunanje vplive nanje identificiramo in kontroliramo. Prvi uspehi znanstvenega pristopa so se pokazali ravno na področju fizike in kemije, kjer pogosto lahko določimo vse zunanje dejavnike, tako da ni nobenih skritih spremenljivk. Analize podatkov potekajo v popolnoma kontroliranem okolju, ponavadi v laboratoriju, kjer so vsi zunanji dejavniki nadzirani in nastavljeni na fiksne vrednosti. Drugače je na področjih biologije, medicine in družbenih ved, kjer je pogosto težko določiti vse pomembne dejavnike, ki vplivajo na interesne spremenljivke.

V splošnem lahko proces znanstvenega raziskovanja prikažemo z naslednjimi koraki:

1. Definiramo raziskovalni problem v obliki znanstvene hipoteze.
2. Zberemo vse relevantne podatke vezane na raziskovalni problem. Upoštevamo vse trenutne informacije o porazdelitvi parametrov matematičnega modela.
3. Oblikujemo anketo, obdelavo ali eksperiment za izvedbo raziskave. Za potrditev ali ovržbo hipoteze mora imeti rezultat raziskave različno vrednost.
4. Naredimo analizo rezultatov in nadgradimo dosedanje znanje o parametrih modela.

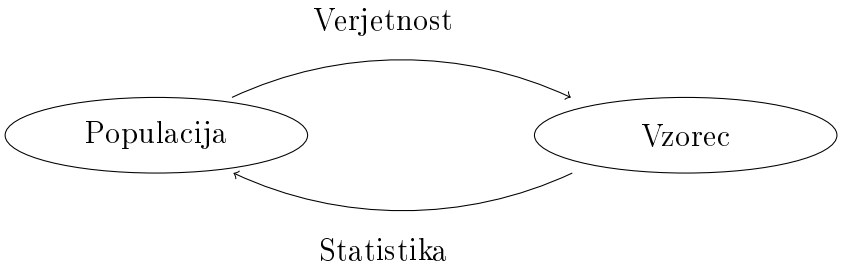
### 2.1.1 Statistično sklepanje

Kadar zunanjih dejavnikov ne moremo določiti, jih tudi ne moremo direktno kontrolirati in zato ne moremo natančno meriti njihovega učinka na spremenljivke v poskusu. Zato lahko pridemo do napačnih sklepov o odnosu med spremenljivkami. V takšnih primerih uporabimo pomembno statistično idejo o *slučajenju* [23] (ang. randomization), ko podatkovne enote slučajno izberemo v skupino, ki jo obravnavamo. S slučajenjem porazdelimo vpliv nedoločljivih zunanjih dejavnikov enakomerno med enote.

*Enota* je posamezni element *populacije*. Populacija je skupek vseh osebkov, objektov in lastnosti, ki nas pri problemu zanimajo. Pogosto je raziskava na celotni populaciji denarno in časovno prezahtevna, na primer raziskava o fizični aktivnosti vseh evropskih prebivalcev starejših od 60 let, ali pa se populacija celo uniči, na primer raziskava o življenjski dobi sveč v izbrani seriji [20]. Pri takšnih pogojih se moramo zadovoljiti samo z delom populacije, ki mu pravimo tudi *vzorec*. Z rezultati raziskave na vzorcu želimo zanesljivo in induktivno sklepati o lastnostih in porazdelitvi izhodne spremenljivke na celotni populaciji.

Porazdelitve opišemo z merami lokacije in merami razpršenosti. Z merami lokacije, kot sta na primer povprečje in mediana, opišemo položaj enot na številčni osi. Z merami razpršenosti, kot so na primer variacijski razpon (ang. range), disperzija in standardni odklon, merimo razpršenost enot. Numeričnim vrednostim mer lokacije in razpršenosti na populaciji pravimo tudi *parametri populacije*. Numerične vrednosti mer lokacije in razpršenosti na vzorcu pa zaznamujemo z besedo *statistike*. Ker je vzorec poznan v celoti, lahko statistike izračunamo. Na podlagi njihovih vrednosti nato sklepamo o vrednostih parametrov populacije. Ker sklepamo iz vzorčnih statistik na neznane populacijske parametre, govorimo o statističnem sklepanju, ki je del tako imenovane *sklepne statistike*.

Verjetnostne porazdelitve določenih značilnosti populacije, ki jo preučujemo, zaznamujejo oblikovanje vzorca. Vzorec deduktivno, a slučajno, oblikujemo na podlagi porazdelitvenih funkcij lastnosti oziroma značilnosti enot na celotni populaciji. Statistika pa nam omogoča induktivno sklepanje o nepoznani porazdelitvi določene lastnosti z vzorca na celotno populacijo [13]. Povezava verjetnosti in statistike s populacijo in vzorcem je grobo orisana na sliki 2.1.




Slika 2.1: Povezava med populacijo in vzorcem preko verjetnosti in statistike.

S slučajnim izborom enot v vzorec dodamo nekaj variabilnosti v podatke. Pri statističnem sklepanju bo vedno nekaj *negotovosti* v rezultatu ravno zaradi variabilnosti podatkov, vendar lahko razvijemo verjetnostni model variabilnosti glede na način slučajenja. Verjetnostni model omogoči oceno negotovosti naših zaključkov. Slučajenje dopusti statistično kontrolo zunanjih dejavnikov, ki jih ne moremo fizično meriti oziroma kontrolirati. Vzorcu, ki ga dobimo s slučajnim izborom, pravimo *enostavni slučajni vzorec* ali na kratko kar *slučajni vzorec*.

Za zgled vzemimo populacijo z $N$ enotami, na kateri preučujemo neko lastnost, ki jo predstavlja slučajna spremenljivka $Y$. Označimo z $Y(e)$ enolično določeno vrednost slučajne spremenljivke $Y$ na poljubni enoti $e$. Naredimo slučajni vzorec velikosti $n$:

$$\boldsymbol{e} = \{e_1, \ldots, e_n\}. \tag{2.1}$$

Z vidika verjetnostnega računa pravimo ugotavljanju vrednosti $Y(e)$, $e \in \boldsymbol{e}$, tudi realizacija slučajne spremenljivke $Y$. Vzorec 2.1 nam tako da $n$ realizacij lastnosti $Y$, recimo

$$\boldsymbol{z} = \{y_1, \ldots, y_n\}, \tag{2.2}$$

kjer je $y_i = Y(e_i)$, za $i = 1, \ldots, n$. To je tudi realizacija slučajnega vektorja

$$\boldsymbol{Z} = \{Y_1, \ldots, Y_n\}. \tag{2.3}$$

Če je vzorec 2.1 slučajen, potem so komponente vektorja 2.3 med seboj neodvisne in vse enako porazdeljene, tako kot je porazdeljena spremenljivka $Y$ na populaciji. Za vzorec lahko vzamemo tako vektor 2.1, kakor tudi vektor 2.3. Neposredno pomembna je le realizacija 2.2 slučajnega vektorja 2.3. Naloga statistike je torej na podlagi končnega števila realizacij slučajne spremenljivke sklepati o njeni neznani porazdelitveni funkciji na celotni populaciji [20].

Za zanesljivo statistično sklepanje mora biti vzorec *reprezentativen*. To pomeni, da vzorec dobro posreduje lastnosti populacije in ohrani razmerja med enotami in porazdelitvami njihovih lastnosti. Slučajni vzorec je eden izmed najbolj učinkovitih načinov, da dobimo reprezentativen vzorec. Večji kot je reprezentativen vzorec, bolj zanesljivo bo naše statistično sklepanje na celotno populacijo.

### 2.1.2 Glavna pristopa k statističnemu sklepanju

Kot je nakazano na sliki 2.1, sta populacija in vzorec tesno povezana s pojmi verjetnostnega računa in statistike. Za obdelavo vzorca in določitev populacijskih parametrov poznamo dva glavna pristopa; frekventistični pristop in Bayesov pristop.

Prvi pristop imenujemo tudi klasični pristop, saj je postavil temelje statističnega sklepanja. Številne statistične metode in strategije sledijo idejam klasičnega pristopa [5], in sicer:

- Populacijski parametri so fiksne, vendar nepoznane konstante.
- Verjetnosti interpretiramo kot limitno vrednost relativne frekvence, ko gre število ponovitev poskusa proti neskončno.
- Učinkovitost statistične metode se oceni na podlagi visokega števila hipotetičnih ponovitev poskusa.

Neznani populacijski parametri so torej fiksno ne slučajno določene konstante, zato ne moremo podati verjetnostnih trditev o njihovih vrednostih. Verjetnostne trditve uporabljamo samo za slučajne vrednosti. Za določitev konstant oblikujemo vzorce

iz populacije, na katerih izračunamo vrednosti statistik. Porazdelitev posamezne statistike na vseh možnih vzorcih določa tako imenovano *porazdelitev vzorčnih ocen* [23]. To porazdelitev v tem delu imenujemo tudi *vzorčna porazdelitev*.

Vzorčna porazdelitev določa območje, kjer se nahaja nek parameter $G$ in ga imenujemo tudi *interval zaupanja*. Vrednost parametra $G$ se nahaja na tem intervalu s *stopnjo zaupanja* $(1-\alpha)$, kadar velja verjetnost $P$:

$$P(g_s \leq G \leq g_z) = 1 - \alpha,$$

kjer je $g_s$ spodnja in $g_z$ zgornja meja tega intervala, $\alpha$ pa *stopnja tveganja* [23].

Torej je statistično sklepanje glede populacijskih parametrov povzeto iz obdelave vseh možnih vzorcev, ki bi jih lahko izbrali za določeno vrednost parametra. Pri dejanskih podatkih in določeni vrednosti parametra tako ni več ničesar slučajnega, zato lahko določamo le intervale zaupanja za določeno vrednost parametra. Ta določitev sloni na vseh vzorcih, ki smo jih in bi jih lahko izbrali.

Nasprotno pa statistično sklepanje pri Bayesovem pristopu temelji samo na obdelavi dejanskih vzorcev, ki smo jih izbrali, in ne na vseh, ki bi jih lahko izbrali, a jih nismo. Parameter obravnavamo kot slučajno spremenljivko, tako da lahko po obdelavi podatkov pogledamo verjetnostne porazdelitve. Osnovne ideje na kratko povzamemo v nekaj točkah [5]:

- Parametre obravnavamo kot slučajne spremenljivke, ker so resnične vrednosti parametrov neznane.
- Pravila verjetnostnega računa so uporabljena direktno pri statističnem sklepanju.
- Verjetnostne trditve interpretiramo kot stopnjo prepričanja, drugače povedano *apriorna porazdelitev* je subjektivna. Apriorna porazdelitev vrednosti parametrov meri kako "verjetna" je posamezna vrednost parametra pred obdelavo podatkov.
- Raziskovalec svoja prepričanja spremeni glede na obdelavo podatkov s pomočjo *Bayesovega izreka*. Nova prepričanja odgovarjajo vrednostim parametrov porazdeljenih po *aposteriorni porazdelitvi*. Aposteriorno porazdelitev dobimo iz dveh virov: apriorne porazdelitve in iz dejanskih podatkov.

Bayesova statistika je tudi napovedna, zato zlahka določimo pogojne porazdelitve naslednje meritve glede na vzorčne podatke. Bayesov pristop omogoča tudi obravnavo tako imenovanih *motečih parametrov* (ang. noise parameters). To so parametri, o katerih ne želimo sklepati, hkrati pa ne želimo, da vplivajo na sklepanje o glavnih parametrih.

## 2.2 Osnove Bayesove statistike

Tradicionalno je statistika prišla do zaključka z uporabo ocen vrednost neznanih spremenljivk v določeni točki. Med njimi je najbolj popularna uporaba koeficien-

tov največjega verjetja. Negotovost ocen se analizira s pomočjo relativnih frekvenc vrednosti ocen in jo določimo z intervalom zaupanja. Frekvenčne vrednosti lahko v večini primerov določimo, kar je izredno uporabno pri statistični analizi. Na drugi strani so pri razvoju Bayesovega pristopa v ospredju celostne verjetnostne porazdelitve vseh neznanih parametrov, ki jih določimo s subjektivno stopnjo prepričanja o apriorni porazdelitvi in z vrednostmi naših opazovanj [28]. Interval zaupanja tukaj nadomestimo z *Bayesovim intervalom*, v katerem se vrednost neznanega parametra nahaja z veliko verjetnostjo.

Pri Bayesovi statistiki je verjetnostna mera temelj za merjenje negotovosti v verjetnostnem modelu. Tukaj lahko obravnavamo tako verjetnost, da ocena vrednosti neznanega parametra leži na določenem intervalu, kakor tudi verjetnost, da povprečje slučajnega vzorca iz znane nespremenljive populacije leži na določenem intervalu. Prva verjetnost je bolj zanimiva po opazovanju, medtem ko je slednja bolj zanimiva a priori, torej pred opazovanjem.

### 2.2.1 Bayesova analiza

Postopek Bayesove analize lahko opišemo v treh korakih [14]:

1. Postavimo verjetnostni model s skupno verjetnostno porazdelitvijo vseh opazovanih in neopazovanih parametrov. Verjetnostni model se mora skladati z našim poznavanjem problema in s postopkom zbiranja podatkov.

2. Na podlagi opazovanj in pogojnih verjetnosti neznanih parametrov glede na dobljene podatke izračunamo in interpretiramo aposteriorne porazdelitve neopazovanih parametrov, ki nas zanimajo.

3. Evalvacija skladnosti verjetnostnega modela s podatki:

   - Ali model ustreza podatkom?
   - Ali dobimo ustrezne rezultate?
   - Kako občutljivi so rezultati na predpostavke modeliranja iz točke 1?

Postavimo verjetnostni model, ki ga določajo parametri modela in podatki. Parametre modela ločimo na opazovane parametre, to so parametri, ki jih lahko opazujemo, in neopazovane parametre, ki predstavljajo sestavni del našega modela, a jih ni mogoče neposredno opazovati. Postavimo oznake parametrov modela in podatkov, ki jih bomo uporabili v nadaljevanju:

- Podatke zaznamujemo z vektorjem $\boldsymbol{y}$.
- Neznane, a opazovane parametre označimo z vektorjem $\tilde{\boldsymbol{y}}$.
- Neopazovane parametre postavimo v vektor $\boldsymbol{\theta}$.

Med parametre spadajo tudi tako imenovane *pojasnjevalne spremenljivke*, ki jih lahko opazujemo, vendar jih ne obravnavamo kot slučajne spremenljivke. Pomembna je še ena predpostavka, in sicer *izmenljivost* $n$ podatkovnih vrednosti $y_i$, kar pomeni,

da je skupna verjetnostna porazdelitev $p(y_1, \ldots, y_n)$ invariantna za permutacijo indeksov. Oznaka $p$ je tukaj generična ne glede na vrsto verjetnostne gostote in hkrati zaznamuje odgovarjajočo porazdelitev.

Verjetnostni model tako predstavimo s skupno verjetnostno gostoto $p(\boldsymbol{\theta}, \boldsymbol{y})$. Statistično sklepanje o neznanih parametrih $\boldsymbol{\theta}$ in še neopazovanih parametrih $\tilde{\boldsymbol{y}}$ je oblikovano na podlagi aposteriornih pogojnih verjetnosti $p(\boldsymbol{\theta}|\boldsymbol{y})$ in $p(\tilde{\boldsymbol{y}}|\boldsymbol{y})$. Skupno verjetnostno gostoto lahko razpišemo s posplošenim pravilom množenja:

$$p(\boldsymbol{\theta}, \boldsymbol{y}) = p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta}),$$

kjer je $p(\boldsymbol{\theta})$ znana marginalna porazdelitev parametrov modela in $p(\boldsymbol{y}|\boldsymbol{\theta})$ pogojna gostota verjetnosti podatkov glede na parametre. Slednji pravimo tudi *porazdelitev podatkov*, medtem ko marginalni porazdelitvi parametrov pravimo tudi *apriorna porazdelitev*.

Iskano aposteriorno gostoto $p(\boldsymbol{\theta}|\boldsymbol{y})$ dobimo z *Bayesovim pravilom*:

$$p(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{p(\boldsymbol{\theta}, \boldsymbol{y})}{p(\boldsymbol{y})} = \frac{p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{y})}, \tag{2.4}$$

kjer je $p(\boldsymbol{y})$ marginalna gostota podatkov, ki jo dobimo iz normalizacijskega pogoja:

$$p(\boldsymbol{y}) = \sum_{\boldsymbol{\theta}} p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta}) \qquad \text{za diskrente slučajne spremenljivke}$$

$$p(\boldsymbol{y}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta})d\boldsymbol{\theta} \qquad \text{za zvezne slučajne spremenljivke}$$

Gostota $p(\boldsymbol{y})$ predstavlja apriorno napovedno porazdelitev (ang. apriori predictive distribution). Apriorna je zato, ker so opazovanja v vzorcu neodvisna od predhodnih opazovanj pri pridobivanju podatkov in napovedna, ker je to porazdelitev parametra, ki ga lahko opazujemo.

S pomočjo Bayesovega pravila lahko tudi napovedujemo vrednosti opazovanih parametrov, le da moramo privzeti, da so napovedni parametri $\tilde{\boldsymbol{y}}$ pogojno neodvisni od podatkov $\boldsymbol{y}$ pri danem $\boldsymbol{\theta}$. Dobimo aposteriorno napovedno porazdelitev, ki je pogojena glede na predhodna opazovanja $\boldsymbol{y}$:

$$p(\tilde{\boldsymbol{y}}|\boldsymbol{y}) = \int p(\tilde{\boldsymbol{y}}, \boldsymbol{\theta}|\boldsymbol{y})d\boldsymbol{\theta} = \int p(\tilde{\boldsymbol{y}}|\boldsymbol{\theta}, \boldsymbol{y})p(\boldsymbol{\theta}|\boldsymbol{y})d\boldsymbol{\theta} = \int p(\tilde{\boldsymbol{y}}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{y})d\boldsymbol{\theta}.$$

Bayesovo pravilo 2.4 predstavimo še v sorazmernostni obliki

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{\theta}),$$

kjer vidimo, da vplivajo podatki na aposteriorno porazdelitev samo preko gostote $p(\boldsymbol{y}|\boldsymbol{\theta})$. Če gledamo porazdelitev podatkov kot funkcijo $\boldsymbol{\theta}$, jo imenujemo tudi *funkcija verjetja*. Kadar imajo na dani množici podatkov modeli isto funkcijo verjetja, dobimo iste aposteriorne porazdelitve. Ta princip opišemo bolj formalno: če obstajata $p(\boldsymbol{y}|\boldsymbol{\theta}) = f(\boldsymbol{\theta})$ in $p(\boldsymbol{y}|\boldsymbol{\theta}) = g(\boldsymbol{\theta})$ ter velja $f = g$, potem sta aposteriorni gostoti enaki, to je $p_f(\boldsymbol{\theta}|\boldsymbol{y}) = p_g(\boldsymbol{\theta}|\boldsymbol{y})$.

**Definicija 2.2.1.** Naj bo $\mathcal{F}$ družina porazdelitev podatkov $p(\boldsymbol{y}|\boldsymbol{\theta})$ in $\mathcal{P}$ družina apriornih porazdelitev $p(\boldsymbol{\theta})$ za $\boldsymbol{\theta}$. Družina $\mathcal{P}$ je *konjugirana* družini $\mathcal{F}$, če velja

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \in \mathcal{P} \quad \forall p(\cdot|\boldsymbol{\theta}) \in \mathcal{F} \text{ in } p(\cdot) \in \mathcal{P}.$$

Najbolj nas zanimajo *naravne* konjugirane porazdelitve, ko družina $\mathcal{P}$ vsebuje samo gostote, ki imajo isto obliko kot funkcija verjetja.

Alternativno lahko prikažemo verjetnost s *kvocienti priložnosti*. Vzemimo dve točki $\theta_1$ in $\theta_2$ ter uporabimo Bayesovo pravilo 2.4:

$$\underbrace{\frac{p(\theta_1|\boldsymbol{y})}{p(\theta_2|\boldsymbol{y})}}_{\substack{\text{kvocient}\\ \text{aposteriornih}\\ \text{priložnosti}}} = \underbrace{\frac{p(\theta_1)}{p(\theta_2)}}_{\substack{\text{kvocient}\\ \text{apriornih}\\ \text{priložnosti}}} \underbrace{\frac{p(\boldsymbol{y}|\theta_1)}{p(\boldsymbol{y}|\theta_2)}}_{\substack{\text{kvocient}\\ \text{verjetij}}}.$$

Drugače povedano, kvocient aposteriornih priložnosti je enak kvocientu apriornih priložnosti pomnoženim s *kvocientom verjetja*.

## 2.3 Gibbsovo vzorčenje

Pri aplikaciji Bayesove analize se v veliki meri uporabljajo *simulacije*, to so konstrukcije verjetnostnih porazdelitev s slučajnimi števili v računalniku [23]. Njihova uporaba je razširjena, saj simulacije omogočajo vzorčenje iz verjetnostnih porazdelitev z relativno lahkoto. Če upoštevamo dualnost med verjetnostno porazdelitvijo in histogramom slučajnih izborov vrednosti iz porazdelitve, potem pri dovolj velikem vzorcu iz histograma povzamemo vse relevantne informacije o gostoti oziroma lahko poračunamo vzorčne statistike. Kontroliramo lahko, kako dobro vzorčne statistike ocenjujejo porazdelitev populacije. Kadar pri simuliranju dobimo ekstremne vrednosti, je to lahko posledica napake pri modeliranju ali parametrizaciji, kar lahko spregledamo pri analitičnem ocenjevanju.

Izbor vrednosti iz verjetnostne porazdelitve sloni na generatorju slučajnih števil. Z dobrim generatorjem slučajnih števil dobimo deterministično zaporedje števil, ki ima iste lastnosti kot zaporedje enakomerno porazdeljenih, slučajno izbranih števil z intervala $[0, 1]$. Pogosta metoda za simuliranje iz diskretnih in zveznih porazdelitev je uporaba kumulativne porazdelitvene funkcije. Kadar poznamo kumulativno porazdelitveno funkcijo $F$ slučajne spremenljivke $X$ in je $F$ strogo naraščajoča, potem za slučajno spremenljivko $U \sim EZ[0, 1]$ velja $F^{-1}(U) \sim F$.

V praksi pogosto simuliramo vrednosti iz aposteriorne porazdelitve $p(\boldsymbol{\theta}|\boldsymbol{y})$ parametrov modela in aposteriorne napovedne porazdelitve $p(\tilde{\boldsymbol{y}}|\boldsymbol{y})$ za neznane opazovane parametre $\tilde{\boldsymbol{y}}$. Simulirane vrednosti $T$ simulacij lahko predstavimo in shranimo v računalniku v obliki urejene tabele 2.1. Posamezno simulirano vrednost indeksiramo z indeksom $t = 1, \ldots, T$ in označimo simulirani vrednosti odgovarjajočega para $(\theta^t, \tilde{y}^t)$, ki ga dobimo iz skupne aposteriorne porazdelitve. Eksponenti v tabeli 2.1 predstavljajo indekse in ne potenc.

Tabela 2.1: Simulirane vrednosti iz aposteriorne in aposteriorno napovedne porazdelitve.

| **Simulacija** | **Parametri** | | | **Opazovani parametri** | | |
|---|---|---|---|---|---|---|
| | $\theta_1$ | $\dots$ | $\theta_k$ | $\tilde{y}_1$ | $\dots$ | $\tilde{y}_n$ |
| 1 | $\theta_1^1$ | $\dots$ | $\theta_k^1$ | $\tilde{y}_1^1$ | $\dots$ | $\tilde{y}_n^1$ |
| $\vdots$ | $\vdots$ | | $\vdots$ | $\vdots$ | | $\vdots$ |
| $T$ | $\theta_1^T$ | $\dots$ | $\theta_k^T$ | $\tilde{y}_1^T$ | $\dots$ | $\tilde{y}_n^T$ |

### 2.3.1 Metode MCMV

Kadar ni mogoče ali je računsko prezahtevno vzorčiti vrednosti parametra $\theta$ direktno iz aposteriorne porazdelitvene funkcije $p(\theta|y)$, potem je ena pogostejših strategij iterativno vzorčenje iz porazdelitvenih funkcij, za katere domnevamo, da so blizu ciljne aposteriorne porazdelitve $p(\theta|y)$. Z vsakim korakom iteracije se tako približamo dejanski porazdelitvi iskanega, a neopazovanega parametra $\theta$.

Gibbsovo vzorčenje spada med metode Monte Carlo Markovske Verige (ang. Monte Carlo Markov Chain), ki temeljijo na iterativnem vzorčenju parametrov $\boldsymbol{\theta}$ iz *ciljne porazdelitve* $p(\boldsymbol{\theta}|\boldsymbol{y})$ (ang. target distribution). Vzorčenje poteka zaporedno po posameznih komponentah vektorja $\boldsymbol{\theta}$. Rezultati trenutnega vzorčenja omogočajo natančnejše vzorčenje pri naslednji iteraciji. Ker pri vsaki iteraciji upoštevamo rezultat predhodne iteracije, formiramo Markovsko verigo, ki pri dovolj velikem številu iteracij konvergira k stacionarni ciljni porazdelitvi $\pi(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{y})$.

#### Metode Monte Carlo

Metode Monte Carlo so se razvile za pomoč pri računanju integralov, ki jih težje izračunamo analitično. Za približek vrednosti integrala uporabimo simulirane vrednosti na podlagi verjetnostnih prepričanj. Poglejmo si enostaven primer Monte Carlo principa.

**Primer 2.3.1.** Delovanje Monte Carlo principa prikažimo s primerom kroga, kvadrata in vrečke riža, [39], kjer želimo oceniti vrednost $\pi = \frac{O}{d}$, ki je razmerje med obsegom kroga $O$ in njegovim premerom $d$.

Narišemo kvadrat z dolžino stranice $d$. Znotraj kvadrata narišemo krog koncentrično, tako da imata krog in kvadrat isti center in je premer kroga enak $d$. Nato vzamemo vrečo riža in zrna riža razporedimo približno enakomerno, a slučajno, znotraj kvadrata $S$. Na koncu preštejemo koliko zrn riža je znotraj kroga, kar označimo s $C$ in koliko zrn je znotraj kvadrata, kar označimo s $S$.

Če smo zrna posuli precej enakomerno, potem bo razmerje med številom zrn v krogu $C$ in številom zrn v kvadratu $S$ približno enako razmerju med ploščinama kroga ter kvadrata in dobimo:

$$\frac{C}{S} \approx \frac{\pi \left(\frac{d}{2}\right)^2}{d^2} \quad \Longrightarrow \quad \pi \approx \frac{4C}{S}.$$

♦

Z zgornjim primerom poenostavljeno prikažemo izračun približne vrednosti integrala. Namreč prava vrednost ploščine kroga $\pi \left(\frac{d}{2}\right)^2$ je enaka vsoti neskončnega števila infinitezimalno majhnih točk. Več zrn riža uporabimo, boljši bo naš približek.

V splošnem se metode Monte Carlo uporabijo za oceno matematičnega upanja neke funkcije $f(z)$ glede na verjetnostno porazdelitev $p(z)$. Spremenljivka $z$ je lahko zvezna ali diskretna. V primeru zvezne spremenljivke želimo oceniti vrednost

$$\mathbb{E}[f] = \int f(z)p(z)dz, \tag{2.5}$$

ki jo ne moremo ali jo težko izračunamo analitično. Glavna ideja je vzorčenje vrednosti $z^t$ za $t = 1, \ldots, T$ neodvisno iz $p(z)$. V zgornjem primeru 2.3.1 so bili naši vzorci zrna riža, s katerimi smo ocenili ploščini kroga in kvadrata. Podobno lahko ocenimo vrednost matematičnega upanja (2.5) s pomočjo končnih delnih vsot, in sicer:

$$\hat{f} = \frac{1}{T} \sum_{t=1}^{T} f\left(z^t\right).$$

Dokler vzorčimo iz $p(z)$, potem velja $\mathbb{E}[\hat{f}] = \mathbb{E}[f]$ in $var[\hat{f}] = \frac{1}{T}\mathbb{E}\left[(f - \mathbb{E}[f])^2\right]$. Enako bi veljalo v primeru večdimenzionalnega vektorja $\boldsymbol{z}$. S takšnim pristopom lahko dosežemo dovolj visoko natančnost z relativno majhnim številom vzorcev.

Eden izmed problemov je, da vzorci $\{z^t\}$ niso nujno neodvisno vzorčeni in imamo zato veliko večje število vzorcev kot jih je potrebno za dobro oceno. Tako je način vzorčenja pomemben glede na podatke, ki jih obravnavamo. Podrobneje o načinih in problemih različnih metod najdemo v literaturi [25], [4].

### Markovska veriga

Pri iterativnem vzorčenju je še posebej zanimiv tako imenovan *Markovski proces*. To je slučajni proces, ki temelji na razvoju *Markovske verige* v času $T$, ki je bodisi disketen $T = \mathbb{N} \cup \{0\}$ bodisi zvezen $T = [0, \infty)$.

**Definicija 2.3.2.** *Slučajni proces* na nekem verjetnostnem prostoru $(\Omega, \mathcal{F}, P)$ je družina slučajnih spremenljivk $\{X_t;\ t \in T\}$.

Nas zanima primer slučajnega procesa, kjer so tako slučajne spremenljivke $X_t$ kot tudi čas $T$ diskretni. Veriga tukaj predstavlja sosledje vrednosti slučajne spremenljivke $X$ v časovnih točkah $t \in T$ glede na verjetnostno mero $P$. Kakšne vrednosti lahko zavzame slučajna spremenljivka, določa diskretna zaloga vrednosti, ki ji pravimo *stanje verige*. Predstavljena je kot števna množica $S$, ki jo tudi imenujemo *množica stanj*. Drugače povedano, za nek slučajni proces $\{X_t\}_{t\in T}$, ki ima množico stanj $S$ kot zalogo vrednosti slučajne spremenljivke $X_t$, $\forall t \in T$, velja, da je $P(X_t \in S) = 1$. Kadar ima veriga še *Markovsko lastnost*, potem govorimo o Markovski verigi.

**Definicija 2.3.3.** Slučajni proces $\{X_{t_i}\}_{t_i\in T}$ ima *Markovsko lastnost*, če za poljuben nabor časovnih točk $t_0 < t_1 < \ldots < t_n$ in poljuben nabor možnih stanj $s_0, s_1, \ldots, s_n$ iz $S$ velja:

$$P\left(X_{t_n} = s_n | X_{t_0} = s_0, X_{t_1} = s_1, \ldots, X_{t_{n-1}} = s_{n-1}\right) = P\left(X_{t_n} = s_n | X_{t_{n-1}} = s_{n-1}\right).$$

Zgornji pogoj pomeni, da je v Markovski verigi zadnje stanje odvisno samo od predhodnega stanja. Tako je dovolj za gradnjo verige začetna točka in tako imenovana *prehodna verjetnost* iz poljubnega stanja $s_i \in S$ v poljubno stanje $s_j \in S$. Začetna točka predstavlja začetno stanje $s_0$ v času $t_0 \in T$, ki naj predstavlja časovno točko, od katere nas proces zanima. To pomeni, da določimo $s_0 \in S$, kjer je $X_{t_0} = s_0$.

Verigo gradimo po *korakih*, ali drugače, iz nekega stanja $s_i$ naredimo korak v drugo stanje $s_j$. Za vsaki poljubni stanji $s_i, s_j \in S$ torej obstaja prehodna verjetnost, ki meri verjetnost prehoda iz stanja $s_i$ v stanje $s_j$ znotraj verige. Formalno jih označimo kot:

$$p_{s_i s_j}(t_j|t_i) = P\left(X_{t_j} = s_j | X_{t_i} = s_i\right).$$

Zapis lahko še poenostavimo, da z indeksi implicitno nakažemo katera stanja nas zanimajo, tako dobimo oznako $p_{ij}(t_j|t_i) = p_{s_i s_j}(t_i|t_j)$.

Pravimo, da je Markovska veriga *homogena*, kadar za vse časovne točke $r, t \in T$, za katere velja $r < t$ in vse $s_i, s_j \in S$ velja:

$$p_{ij}(t|r) = p_{ij}(t - r|0). \tag{2.6}$$

Zgornja enačba 2.6 pove, da je prehodna verjetnost $p_{ij}(t|r)$ neodvisna od $r$. S tem pogojem gradnja Markove verige poteka iz vsake časovne točke enako. V tem primeru lahko poenostavljeno zapišemo:

$$p_{ij}(t) := p_{ij}(t|0), \quad s_i, s_j \in S,\ t \in T.$$

Naj bo $p_{ij} = p_{ij}(1)$ prehodna verjetnost iz stanja $s_i$ v stanje $s_j$ na prvem koraku homogene verige. Vse prehodne verjetnosti lahko zapišemo v *prehodno matriko* $P = (p_{ij})_{s_i,s_j\in S}$. Prehodna matrika homogene Markovske verige z diskretnim časom je stohastična in zanjo velja:

1. $p_{ij} \geq 0$,

2. $\sum_{s_j \in S} p_{ij} = \sum_{s_j \in S} P(X_1 = s_j | X_0 = s_i) = 1.$

Za korak $t$ imamo prehodno matriko $P(t) = (p_{ij}(t))$, za katero velja $P(t) = P^t$. Kadar so elementi prehodne matrike $P(t)$ pozitivni za poljuben $t$, potem je vsako stanje $s_i$ *dosegljivo* iz stanja $s_j$ po $t$ korakih s pozitivno verjetnostjo. V tem primeru pravimo, da je Markovska veriga nerazcepna. Kadar nobene podmnožice stanj v $S$ veriga ne obiskuje ciklično, potem je veriga neperiodična.

Za stanje $s_i \in S$ označimo *čas prve vrnitve* s $T_i$ in velja:

$$T_i = k \quad \Leftrightarrow \quad X_k = s_i | X_0 = s_i, X_1 \neq s_i, \ldots, X_{k-1} \neq s_i.$$

V primeru, da se veriga nikoli ne vrne v stanje $s_i$, je $T_i = \infty$ in pravimo, da je stanje $s_i$ minljivo. Drugače je stanje $s_i$ povrnljivo ali ponovljivo (ang. recurrent). Če je $T_i \in L^1$, potem je stanje povrnljivo neničelno, v nasprotnem pa je povrnljivo ničelno. Povrnljivo neničelna neperiodična stanja imenujemo tudi *ergodijska*.

Označimo začetno porazdelitev stanj verige s $p(0) = (p_1(0), p_2(0), \ldots, p_n(0))$ za nek $n \in \mathbb{N}$. Zanima nas porazdelitev $p(n) = p(0) \cdot P^n$, ko gre $n$ proti neskončno. Kadar obstaja porazdelitev $\pi = (\pi_1, \pi_2, \ldots)$ z lastnostmi:

- $\pi_i \geq 0$,
- $\sum_{s_i \in S} \pi_i = 1$,
- $\pi P = \pi$,

ji pravimo tudi *stacionarna porazdelitev*. Nerazcepna veriga ima enolično stacionarno porazdelitev natanko tedaj, kadar so vsa stanja povrnljivo neničelna.

*Simulacija z Markovsko verigo* se začne v slučajno ali deterministično izbrani začetni točki $t_0$ in z neodvisnimi sekvenčnimi koraki zgradi verigo slučajnih spremenljivk, katere gradnja je določena s prehodnimi verjetnostmi. Porazdelitev prehodnih verjetnosti mora omogočiti konvergenco Markovske verige k enolični stacionarni porazdelitvi $\pi(\theta)$.

### 2.3.2 Gibbsov vzorčevalnik

Gibbsov vzorčevalnik je najpreprostejši primer Monte Carlo metode, ki uporabi simulacijo z Markovsko verigo. V posebnem je to različica algoritma *Metropolis-Hastings*. Najprej opišemo delovanje algoritma *Metropolis-Hastings*, nato pa predstavimo Gibbsov vzorčevalnik. Poglavje zaključimo s krajšo razlago o konvergenci dobljene Markovske verige k stacionarni porazdelitvi.

### Algoritem *Metropolis-Hastings*

Algoritem *Metropolis-Hastings* je posplošitev algoritma Metropolis, po katerem s tako imenovanim sprejemnim pogojem (ang. acceptance rate) zgradimo Markovsko verigo, da konvergira k ciljni porazdelitvi. Tukaj na kratko predstavimo algoritem *Metropolis-Hastings*, ki je podlaga za Gibbsov vzorčevalnik. Več o algoritmu Metropolis pa si lahko bralec prebere v delu [14].

Predpostavimo torej, da želimo vzorčiti iz večdimenzionalne porazdelitvene funkcije $p(\boldsymbol{\theta}|\boldsymbol{y})$, kjer vektor $\boldsymbol{y}$ predstavlja znane podatke. Ker bomo gradili verigo, moramo določiti začetno točko $\boldsymbol{\theta}^0$. Začetno točko izberemo iz *začetne porazdelitve* $p_0(\boldsymbol{\theta})$ (ang. starting distribution), ki predstavlja apriorno porazdelitev $\boldsymbol{\theta}$ in je tako subjektivne narave. Pogosto začetno porazdelitev ocenimo po neki drugi metodi. Začetna točka $\boldsymbol{\theta}^0$ predstavlja začetno stanje verige in mora ustrezati $p\left(\boldsymbol{\theta}^0|\boldsymbol{y}\right) > 0$. Eksponenti predstavljajo časovne točke enako kot pri tabeli 2.1 in ne potenc.

Za nadaljnje vzorčenje pa potrebujemo še tako imenovano *predlagano porazdelitev* $J_t\left(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{t-1}\right)$ (ang. proposal distribution, tudi jumping distribution). Indeks $t$ predstavlja časovno točko in stanje $\boldsymbol{\theta}^*$ predstavlja predlog za naslednje stanje v verigi pri koraku $t$. Pri algoritmu Metropolis je predlagana porazdelitev simetrična, kar pa ni več potrebno pri algoritmu *Metropolis-Hastings*. Asimetrija omogoča pospeševanje konvergence.

Glede na *sprejemni pogoj*

$$\alpha = \frac{p\left(\boldsymbol{\theta}^*|\boldsymbol{y}\right)/J_t\left(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{t-1}\right)}{p\left(\boldsymbol{\theta}^{t-1}|\boldsymbol{y}\right)/J_t\left(\boldsymbol{\theta}^{t-1}|\boldsymbol{\theta}^*\right)}$$

se odločimo, ali bomo predlagano stanje $\boldsymbol{\theta}^*$ sprejeli z verjetnostjo $\min(\alpha, 1)$ ali pa zavrnili in predlagali novo stanje. Sprejemni pogoj je vedno definiran, ker se korak iz stanja $\boldsymbol{\theta}^{t-1}$ v stanje $\boldsymbol{\theta}^*$ zgodi samo, kadar sta obe vrednosti $p\left(\boldsymbol{\theta}^{t-1}|\boldsymbol{y}\right)$ in $J_t\left(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{t-1}\right)$ različni od nič. Psevdokoda za algoritem *Metropolis-Hastings* je zapisana v algoritmu 1.

Dobimo tako imenovano *Metropolis-Hastings* Markovsko verigo $\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^T$, ki je približno porazdeljena kot $f(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{y})$ za velike vrednosti $T$. Naj bo $\boldsymbol{x} = \boldsymbol{\theta}^{t-1}$ in $\boldsymbol{z} = \boldsymbol{\theta}^t$. Potem lahko iteracijo v algoritmu enačimo z vzorčenjem iz *prehodne funkcije*

$$\kappa(\boldsymbol{z}|\boldsymbol{x}) = \alpha \cdot J_t(\boldsymbol{z}|\boldsymbol{x}) + (1 - r(\boldsymbol{x}))\,\delta_{\boldsymbol{x}}(\boldsymbol{z}),$$

kjer je $r(\boldsymbol{x}) = \int \alpha \cdot J_t(\boldsymbol{z}|\boldsymbol{x})\, d\boldsymbol{z}$ in $\delta_{\boldsymbol{x}}(\boldsymbol{z})$ oznaka za Diracovo delta funkcijo. Prehodna funkcija zadošča *pogoju simetričnega ravnotežja*:

$$f(\boldsymbol{x})\kappa(\boldsymbol{y}|\boldsymbol{x}) = f(\boldsymbol{y})\kappa(\boldsymbol{x}|\boldsymbol{y}), \tag{2.7}$$

iz katerega sledi, da je $f$ stacionarna porazdelitev verige. Pri nekaj dodatnih predpostavkah, je $f$ tudi limitna porazdelitev verige [25].

**Algoritem 1** Psevdokoda algoritma *Metropolis-Hastings*.

**Vhod:** Začetna točka $\boldsymbol{\theta}^0$, gostota $p(\boldsymbol{\theta}|\boldsymbol{y})$ in predlagana porazdelitev $J$.
**Izhod:** Slučajni vzorec $\left\{\boldsymbol{\theta}^t\right\}_{t=1}^{T}$ s stacionarno porazdelitvijo $\pi(\boldsymbol{\theta})$.

1: **while** $t = 1 : T$ ali konvergenca **do**
2: Predlagaj naslednje stanje $\boldsymbol{\theta}^*$ iz porazdelitve $J_t\left(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{t-1}\right)$.
3: Izračunaj sprejemni pogoj

$$\alpha = \frac{p\left(\boldsymbol{\theta}^*|\boldsymbol{y}\right)/J_t\left(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{t-1}\right)}{p\left(\boldsymbol{\theta}^{t-1}|\boldsymbol{y}\right)/J_t\left(\boldsymbol{\theta}^{t-1}|\boldsymbol{\theta}^*\right)}.$$

4: Postavi

$$\boldsymbol{\theta}^t = \begin{cases} \boldsymbol{\theta}^* & \text{z verjetnostjo } \min(\alpha, 1) \\ \boldsymbol{\theta}^{t-1} & \text{sicer} \end{cases}.$$

5: **end while**

**vrni** $\left\{\boldsymbol{\theta}^T\right\} \sim \pi(\boldsymbol{\theta})$.

### Gibbsov algoritem

Naj bodo naši neznani parametri $\theta_1, \ldots, \theta_n$ za poljuben $n \in \mathbb{N}$ zbrani v vektor $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^n$. Pri Gibbsovem vzorčenju delamo s komponentami, ki predstavljajo podvektorje vektorja $\boldsymbol{\theta}$. Znotraj vsake iteracije $t$ naredi Gibbsov vzorčevalnik toliko korakov, kolikor je komponent. Pri vsaki iteraciji $t$ vzorčimo $j$-to komponento $\theta_j$ vektorja $\boldsymbol{\theta}$ iz pogojne porazdelitve, in sicer:

$$\theta_j^t \sim p\left(\theta_j|\theta_{-j}^{t-1}, y\right),$$

kjer velja $\theta_{-j}^{t-1} = \left(\theta_1^t, \ldots, \theta_{j-1}^t, \theta_{j+1}^{t-1}, \ldots, \theta_n^{t-1}\right)$. Iz zapisa $\theta_{-j}^{t-1}$ razberemo, da pri $j$-tem koraku iteracije $t$ že uporabimo nove vrednosti komponent $\theta_i^t$ za $i < j$. Na ta način uporabimo vse informacije, ki so na voljo do trenutnega koraka.

Z Gibbsovim vzorčevalnikom verjetnostno ne določimo naslednjega stanja $t$ za vse komponente $\boldsymbol{\theta}$ hkrati, ampak naredimo ločeno verjetnostno odločitev za vsako od $n$ komponent, kjer je vsaka odločitev odvisna od predhodnih $n-1$ komponent [39]. Verjetnostna odločitev tukaj predstavlja skok iz enega stanja komponente v drugo stanje komponente pri trenutni iteraciji. Za začetek iteracije izberemo začetno točko $\boldsymbol{\theta}^0$ iz porazdelitve $p(\boldsymbol{\theta})$. Psevdokoda Gibbsovega vzorčevalnika je opisana v algoritmu 2.

Za razliko od algoritma *Metropolis-Hastings* tukaj ni potrebno poračunati sprejemnega pogoja, saj je zaradi izbora predlagane porazdelitve $J$, vrednost pogoja $\alpha \equiv 1$. Porazdelitev $J$ prav tako gledamo po komponentah in jo za komponento $j$ v koraku $t$ definiramo kot

$$J_{j,t}\left(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{t-1}\right) = \begin{cases} p\left(\theta_j^*|\boldsymbol{\theta}_{-j}^{t-1}, \boldsymbol{y}\right) & \text{če } \boldsymbol{\theta}_{-j}^* = \boldsymbol{\theta}_{-j}^{t-1} \\ 0 & \text{sicer} \end{cases}.$$

**Algoritem 2** Psevdokoda Gibbsovega vzorčenja.

**Vhod:** Začetna točka $\boldsymbol{\theta}^0$ in gostota $p(\boldsymbol{\theta}|\boldsymbol{y})$.
**Izhod:** Slučajni vzorec $\left\{\boldsymbol{\theta}^t\right\}_{t=1}^{T}$ s stacionarno porazdelitvijo $\pi(\boldsymbol{\theta})$.

1: **while** $t = 1 : T$ ali konvergenca **do**
2: **for** $j = 1 : n$ **do**
3: $\theta_j^t \sim p\left(\theta_j|\theta_{-j}^{t-1}, \boldsymbol{y}\right)$
4: **end for**
5: **end while**

**vrni** $\left\{\boldsymbol{\theta}^T\right\} \sim \pi(\boldsymbol{\theta})$.

Zato pri vsaki iteraciji sprejmemo predlog $\boldsymbol{\theta}^*$, saj velja:

$$\begin{aligned}\alpha &= \frac{p\left(\boldsymbol{\theta}^*|\boldsymbol{y}\right)/J_{j,t}\left(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{t-1}\right)}{p\left(\boldsymbol{\theta}^{t-1}|\boldsymbol{y}\right)/J_{j,t}(\boldsymbol{\theta}^{t-1}|\boldsymbol{\theta}^*)} \\ &= \frac{p\left(\boldsymbol{\theta}^*|\boldsymbol{y}\right)/p\left(\theta_j^*|\boldsymbol{\theta}_{-j}^{t-1}, \boldsymbol{y}\right)}{p\left(\boldsymbol{\theta}^{t-1}|\boldsymbol{y}\right)/p\left(\theta_j^{t-1}|\boldsymbol{\theta}_{-j}^{t-1}, \boldsymbol{y}\right)} \\ &= \frac{p\left(\boldsymbol{\theta}_{-j}^{t-1}|\boldsymbol{y}\right)}{p\left(\boldsymbol{\theta}_{-j}^{t-1}|\boldsymbol{y}\right)} \\ &\equiv 1.\end{aligned}$$

### Konvergenca algoritmov

Algoritem 2 predstavlja sistematično različico Gibbsovega vzorčenja. Tukaj komponente zaporedno nadgradimo, torej vzorčenje komponent poteka v vrstnem redu

$$1 \to 2 \to \ldots \to n.$$

Naj bo $f(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\boldsymbol{y})$. Prehodno verjetnost z oznakami iz podpoglavja 2.3.1 za stanji $\boldsymbol{x} = \boldsymbol{\theta}^{t-1}$ in $\boldsymbol{z} = \boldsymbol{\theta}^t$ zapišemo kot

$$p_{xz}(t|t-1) = \prod_{j=1}^{n} f(z_j|z_1, \ldots, z_{j-1}, x_{j+1}, \ldots, x_n) = \prod_{j=1}^{n} f(z_j|x_{-j}).$$

Da poudarimo zaporedje vzorčenja komponent, označimo prehodno funkcijo kot $\kappa_{1\to n}(z|x) = p_{xz}(t|t-1)$. Tehnična podlaga za konvergenco je povzeta z izrekom (Hammersley-Clifford).

**Izrek 2.3.4.** *Naj bo $f(x_i)$ i-ta marginalna gostota porazdelitve $f(\boldsymbol{x})$. Naj gostota $f(\boldsymbol{x})$ zadosti pogoju pozitivnosti, torej za vsak $\boldsymbol{z} \in \{\boldsymbol{x} : f(x_i) > 0, i = 1, \ldots, n\}$ velja $f(\boldsymbol{z}) > 0$. Potem velja*

$$f(\boldsymbol{z})\kappa_{n\to 1}(\boldsymbol{x}|\boldsymbol{z}) = f(\boldsymbol{x})\kappa_{1\to n}(\boldsymbol{z}|\boldsymbol{x}). \tag{2.8}$$

Z integracijo pogoja Hammerlsey-Clifford 2.8 na obeh straneh po spremenljivkah $\boldsymbol{x}$ dobimo globalni pogoj simetričnega ravnotežja:

$$\int f(\boldsymbol{x})\kappa_{1\to n}(\boldsymbol{z}|\boldsymbol{x})d\boldsymbol{z} = f(\boldsymbol{z}).$$

Podobno kot pri pogoju 2.7 iz pogoja 2.8 sledi, da je $f$ stacionarna porazdelitev Markovske verige s prehodno funkcijo $\kappa_{1\to n}(\boldsymbol{z}|\boldsymbol{x})$. To velja, če je Markovska veriga nerazcepna, neperiodična in nima minljivih stanj.

Pogoj pozitivnosti na funkcijo $f$ implicira, da je Markovska veriga Gibbsovega algoritma nerazcepna in da je $f$ njena limitna funkcija. Za večino poznanih porazdelitev, s katerimi gradimo Gibbsovo Markovsko verigo velja, da je veriga neperiodična in nima minljivih stanj. Za podrobnejšo teoretično razlago in primere, naj si bralec ogleda deli [25] in [14].

## 2.4 O grafih

Teorija grafov je področje matematike, ki se ukvarja s proučevanjem lastnosti grafov in je v splošnem precej široko. Privlačen vidik študija teorije grafov je prav njena uporabnost pri reševanju realnih problemov. Sam pojem grafa je bil razvit za reševanje realnega problema mostov v Königsbergu, ki ga je razrešil Euler leta 1736 [24]. Grafi nam omogočijo, da orišemo in strukturiramo dane podatke. Nadaljnja analiza dobljenega grafa lahko izpostavi določene odvisnosti med podatki in njihove inherentne lastnosti.

V tem podpoglavju predstavimo nekaj osnovnih pojmov iz teorije grafov [37], [35], ki smo jih uporabili za razvoj naše rešitve pri iskanju motivov v DNA sekvencah. Prikažemo še algoritem *Iskanje v širino* (ang. Breadth-first search algorithm), ki smo ga prilagodili za potrebe našega algoritma *GraphGibbs*.

### 2.4.1 Osnovni pojmi

*Graf* $G$ je urejen par $(V, E)$, kjer je $V = V(G)$ neprazna množica *vozlišč* in $E = E(G)$ množica *povezav*. Povezava $e \in E$ je predstavljena s parom vozlišč $u, v \in V(G)$, tako da velja $e = \{u, v\}$. Vozliščema $u$ in $v$ v $e = \{u, v\}$ pravimo tudi *krajišči* povezave $e$. Kadar med poljubnima vozliščema $u, v \in V(G)$ obstaja povezava, potem sta si vozlišči *sosednji*, kar označimo kot $u \sim v$, kjer $\sim$ predstavlja *relacijo sosednosti*. Povezavi sta si sosednji, če imata skupno vozlišče.

Relacija sosednosti $R_\sim$ je dvomestna relacija v $V(G)$ in jo lahko predstavimo kot množico urejenih parov elementov iz $V(G)$. Drugače povedano velja $R_\sim \subseteq V(G) \times V(G)$. Kadar je relacija $R_\sim$ simetrična, pomeni, da lahko povezave predstavimo kot neurejene pare in pravimo, da je graf *neusmerjen*. Če je relacija $R_\sim$ asimetrična, potem povezave definiramo kot urejene pare ali *loke* oblike $(u, v)$, kjer sta si vozlišči $u, v \in V(G)$ sosednji. Tako vsaka povezava $e = \{u, v\}$ definira dva med seboj nasprotna loka $(u, v)$ in $(v, u)$. V tem primeru govorimo o *usmerjenem grafu*. Pri

lokih ločimo tudi *začetno* in *končno* krajišče.

Če je relacija $R_\sim$ poleg simetričnosti še irefleksivna in ni večkratna množica, potem dobimo *enostavni graf*. Irefleksivnost pomeni, da nobeno vozlišče $u \in V(G)$ ni v relaciji samo s seboj. Torej graf nima *zank* oziroma povezav, ki bi imele obe krajišči v istem vozlišču. Pojem enostavni graf prav tako ne dopušča možnosti *večkratnih povezav*. Večkratna povezava ali tudi vzporedna povezava predstavlja več različnih povezav, ki imajo za svoja krajišča isti par vozlišč. V primeru, ko ima graf zanke in dopušča večkratne povezave, govorimo o *multigrafu*. Tako kot enostavni graf je tudi multigraf lahko usmerjen ali neusmerjen.

Graf $G$ je *končen*, kadar sta množici $V(G)$ in $E(G)$ končni. Moč množice $V(G)$ določa tako imenovani *red grafa*. Za vsako vozlišče $u \in V(G)$ lahko definiramo množico vseh njegovih sosedov v grafu $G$, ki jo imenujemo *okolica vozlišča* in označimo z $N(u)$. Njena moč pa predstavlja *stopnjo* ali *valenco* $val(u)$ vozlišča $u$.V kolikor je graf usmerjen, potem ločimo tudi med vhodno $val^+$ in izhodno $val^-$ stopnjo vozlišča, saj je lahko vozlišče bodisi začetno bodisi končno krajišče povezave. Podobno lahko v usmerjenem grafu označimo sosede vozlišča $u$ pri vseh izhodnih povezavah z $N^-(u)$ in njegove sosede pri vseh vhodnih povezavah ozriroma lokih kot $N^+(u)$.

V grafu $G$ lahko tudi označimo vozlišča in povezave z znaki iz izbrane abecede. Kadar v grafu označimo vozlišča, dobimo *označen graf*. Če pa numerično označimo povezave, torej jih utežimo, pa dobimo *utežen graf*. Utežen in v večini primerov usmerjen graf se imenuje tudi *omrežje*.

*Podgraf* $H$ grafa $G$ je graf na podmnožici vozlišč in povezav iz $G$. Na podmnožici $U \subseteq V(G)$ lahko definiramo *induciran podgraf* $H[U] = (U, E(U) \cap E(G))$ grafa $G$, ki ima za povezave vse tiste povezave iz $E(G)$, ki imajo obe krajišči v množici $U$.

*Sprehod* $W$ dolžine $n \geq 0$ od vozlišča $u = u_1$ do vozlišča $v = u_{n+1}$ je izmenično zaporedje vozlišč in povezav $W = (u_1, e_{12}, u_2, e_{23}, \ldots, e_{n(n+1)}, u_{n+1})$, kjer je $e_{i(i+1)} = (u_i, u_{i+1})$ za vse $i \in \{1, \ldots, n\}$. Označimo ga tudi kot $W : u \xrightarrow{n} v$. V primeru enostavnega grafa lahko sprehod gledamo poenostavljeno kot zaporedje sosednjih vozlišč $W = (u_1, u_2, \ldots, u_{n+1})$. *Dolžino* sprehoda merimo s številom povezav v sprehodu. Zaporedje $(u)$ je *trivialen sprehod* dolžine 0 v vozlišču $u$. Če sprehod vsebuje le različna vozlišča, potem mu pravimo tudi *pot*. Tako je lok ravno pot dolžine 1.

Graf $G$ reda $r$ lahko predstavimo na več načinov [24], med drugim z *matriko sosednosti* ali s *seznamom sosedov*. Matrika sosednosti je kvadratna matrika $A(G) = [a_{ij}]$ dimenzij $r \times r$, kjer so elementi določeni po formuli:

$$a_{ij} = \begin{cases} 1; & i \sim j \\ 0; & \text{sicer.} \end{cases}$$

Za neusmerjen graf je matrika sosednosti $A(G)$ simetrična. V splošnem velja:

$$\sum_{j=1}^{r} a_{ji} = \text{vhodna stopnja } i$$
$$\sum_{j=1}^{r} a_{ij} = \text{izhodna stopnja } i$$

Seznam sosedov vodimo z dvema strukturama, in sicer strukturo *začetek*, ki za vsako vozlišče pove, kje se seznam sosedov začne, in s strukturo *sosedi*, ki za vsako vozlišče našteje vse sosede. Obe strukturi pogosto vodimo v tabeli.

### 2.4.2 Algoritem: *Iskanje v širino*

*Pregled* grafa poteka, dokler ne obiščemo vseh vozlišč ali pa ne najdemo vozlišča z izbrano lastnostjo. Vse operacije, ki jih izvršimo na vozlišču, povzamemo s pojmom *obisk vozlišča*. Poznamo dva glavna načina za pregled grafa, in sicer algoritem *Iskanje v globino* (ang. Depth-first search) ter algoritem *Iskanje v širino*. Z obema pristopoma naredimo pregled grafa, a obiščemo vozlišča v drugačnem vrstnem redu.

Algoritem *Iskanje v širino* (BFS) smo z manjšo spremembo uporabili tudi v algoritmu *GraphGibbs*, zato si postopek poglejmo bolj natančno. Ime *Iskanje v širino* pride iz dejstva, da vse sosede trenutnega vozlišča $v$, ki še niso bili obiskani, takoj uvrstimo med kandidate za naslednji obisk. Izbor naslednjega vozlišča iz množice kandidatov pa izberemo po principu "kdor prej pride, prej melje".

Torej za graf $G = (V, E)$ in izbrano vozlišče $s \in V(G)$, ki mu rečemo tudi *koren*, algoritem BFS sistematično obišče vsa vozlišča, ki so *dosegljiva* iz vozlišča $s$. Vozlišče $u \in V(G)$ je dosegljivo iz $s$, če obstaja pot med vozliščema $s$ in $u$. Algoritem BFS sproti izračuna *razdaljo* $\delta(s, v)$, ki v tem primeru zaznamuje najmanjše število povezav iz $s$ do vsakega dosegljivega vozlišča $v \in V(G)$. Prav tako oblikuje drevo s korenom $s$, ki vsebuje vsa dosegljiva vozlišča iz $s$.

Pri pregledu grafa $G$ vodimo logičen seznam *obiskani*, kjer sproti beležimo obiskana vozlišča. Na začetku nobeno vozlišče ni obiskano, torej so vrednosti v seznamu *obiskani* nastavljene na vrednost *false*. Prav tako za vsako vozlišče $v \in V(G)$ sledimo njegovi razdalji $\delta(v) = \delta(s, v)$ iz korena $s \in V(G)$. Pred zagonom algoritma BFS so razdalje $\delta(s, v)$ za vsa vozlišča $v$ nastavljene na vrednost $\infty$. S tem povemo, da še nismo zgradili poti iz korena $s$ do vozlišča $v$. Drevo, ki ga algoritem BFS gradi ob pregledu, predstavimo s seznamom $\pi(\cdot)$ prvega prednika ali očeta vsakega obiskanega vozlišča $v$. Tako $\pi(v) = u$ pomeni, da je vozlišče $u$ oče vozlišča $v$ v drevesu. Ker je vsako vozlišče grafa obiskano največ enkrat, ima tudi vsako vozlišče drevesa največ enega predhodnika (očeta). Z oznako $\pi(v) = \text{NIL}$ povemo, da je $v = s$ ali da vozlišče $v$ še ni bilo obiskano.

Algoritem BFS deluje tako na usmerjenih kot na neusmerjenih grafih. Predpostavljamo, da je graf $G = (V, E)$ predstavljen s seznamom sosedov $N(\cdot)$. Podan

imamo tudi koren $s \in V(G)$. Poleg seznama razdalj $\delta(v)$ in seznama očetov $\pi(v)$ za vsako vozlišče $v \in V(G)$, vodimo tudi urejen seznam (vrsto) $vrsta$, ki vsebuje kandidate za obisk. Psevdokoda je zapisana v algoritmu 3.

**Algoritem 3** Psevdokoda algoritma *Iskanje v širino*.

**Vhod:** Seznam sosedov $N(\cdot)$ za vsa vozlišča iz $V(G)$ in koren $s \in V(G)$.
**Izhod:** Seznam razdalj $\delta(s, v)$ in seznam očetov $\pi(v)$ za vsako vozlišče $v \in V(G)$.

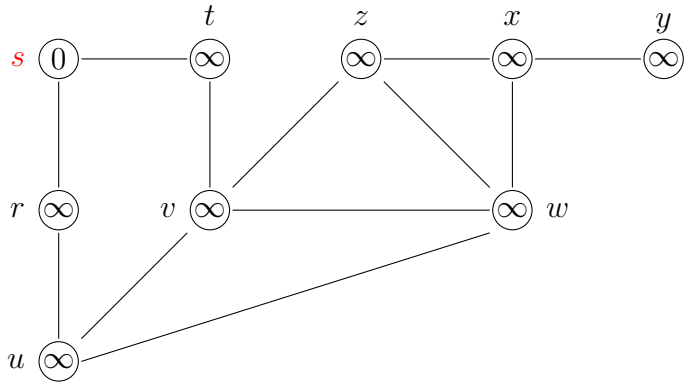
```
1: for all vozlišče u ∈ V(G) − {s} do
2:    obiskani(u) = false
3:    δ(u) = ∞
4:    π(u) = NIL
5: end for
6: obiskani(s) = true
7: δ(s) = 0
8: π(s) = NIL
9: Vstavi (vrsta, s).
10: while vrsta ≠ ∅ do
11:    Briši (vrsta, u).
12:    for all v ∈ N(u) do
13:       Obišči v.
14:       if obiskani(v) = false then
15:          obiskani(v) = true
16:          δ(v) = δ(u) + 1
17:          π(v) = u
18:          Vstavi (vrsta, v)
19:       end if
20:    end for
21: end while
```

**vrni** seznam razdalj $\delta(\cdot)$ in drevo $\pi(\cdot)$

Algoritem BFS torej izračuna najkrajše poti iz korena $s$ do vsakega dosegljivega vozlišča $v$ in sestavi drevo pri pregledu grafa. Algoritem deluje pravilno, kar lahko povzamemo z izrekom 2.4.1. Dokaz si lahko bralec ogleda v delih [24] in [53], vendar si za boljšo predstavo psevdokode algoritma BFS v algoritmu 3 le poglejmo delovanje algoritma BFS na primeru 2.4.2, s katerim lahko tudi vizualno potrdimo izrek 2.4.1.

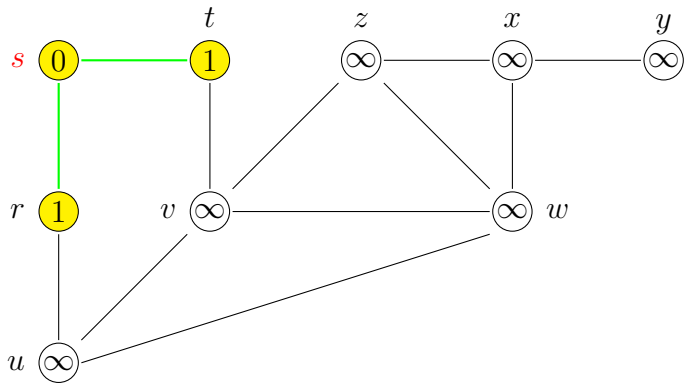
**Izrek 2.4.1.** *Naj bo $G = (V, E)$ usmerjen ali neusmerjen graf in vozlišče $s \in V(G)$ koren. Potem algoritem BFS med izvajanjem odkrije vsako vozlišče $v \in V(G)$, ki je dosegljivo iz korena $s$. Za izhod poda seznam najkrajših poti $\delta(s, v)$ za vsa vozlišča $v$ in drevo $\pi(\cdot)$. Dodatno: za vsako vozlišče $v \neq s$, ki je dosegljivo iz $s$, je ena izmed najkrajših poti iz $s$ do $v$ ravno najkrajša pot iz $s$ do $\pi(v)$, ki ji dodamo povezavo $(\pi(v), v)$.*

**Primer 2.4.2.** Poglejmo si delovanje algoritma BFS na enostavnem neusmerjenem grafu $G$:




Znotraj vozlišč beležimo razdalje posameznih vozlišč iz korena $s$. Naravno sledi, da je $\delta(s) = \delta(s, s) = 0$. Ostalih vozlišč še nismo obiskali, torej je pot iz $s$ do vsakega vozlišča neskončna oziroma ne obstaja. Sproti bomo vodili še *vrsto* in seznam očetov $\pi$ v trenutnem drevesu. Na začetku postavimo v vrsto koren $s$ in označimo $\pi(s) = \text{NIL}$. Sosednja vozlišča obiščemo v slučajnem vrstnem redu in povezave, ki jih uporabimo označimo z zeleno barvo.

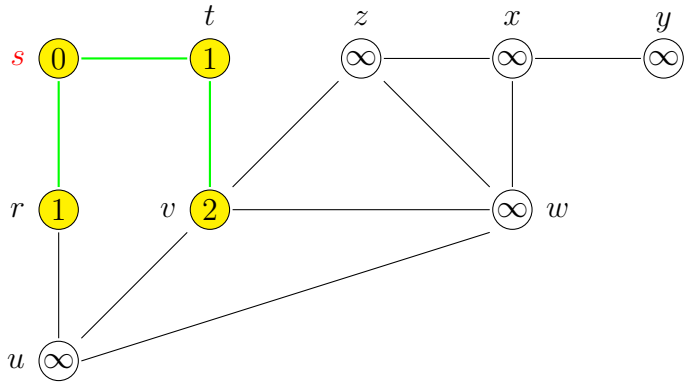
Korak 1. Obiščemo vozlišči $t$ in $r$. Velja $\delta(s, t) = \delta(s, r) = 1$ in $\pi(t) = \pi(r) = s$.




$vrsta: \{t, r\}$

$\pi: \{s, s\}$

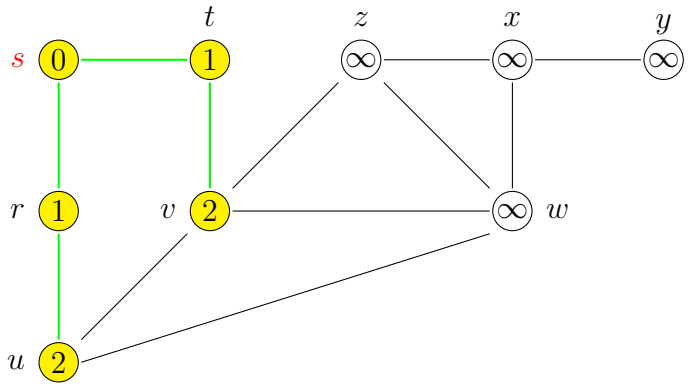
Korak 2. Obiščemo vozlišče $v$. Velja $\delta(s, v) = 2$ in $\pi(v) = t$.




$vrsta: \{r, v\}$

$\pi: \{s, t\}$

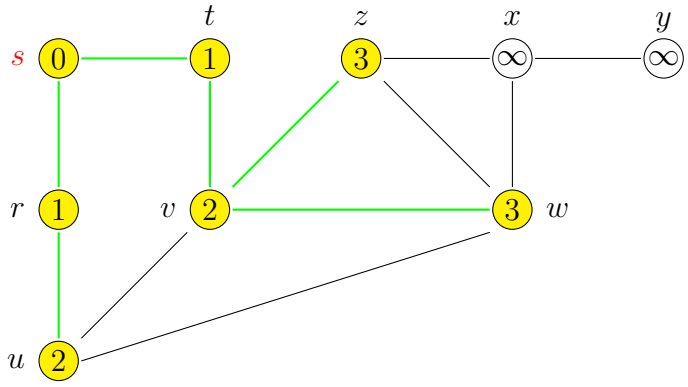
Korak 3. Obiščemo vozlišče $u$. Velja $\delta(s,u) = 2$ in $\pi(u) = r$.




$vrsta: \{v, u\}$

$\pi: \{t, r\}$

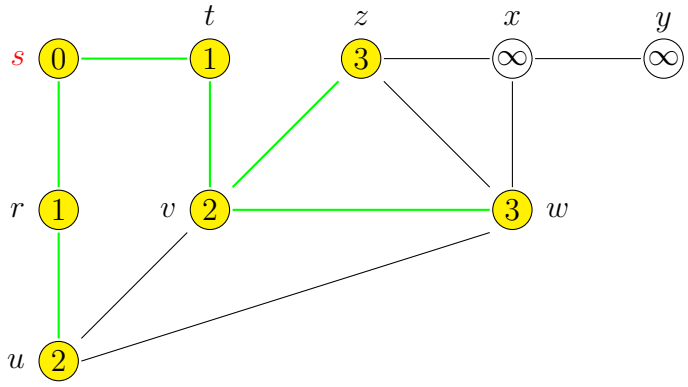
Korak 4. Obiščemo vozlišči $z$ in $w$. Velja $\delta(s,z) = \delta(s,w) = 3$ in $\pi(z) = \pi(w) = v$.




$vrsta: \{u, z, w\}$

$\pi: \{r, v, v\}$

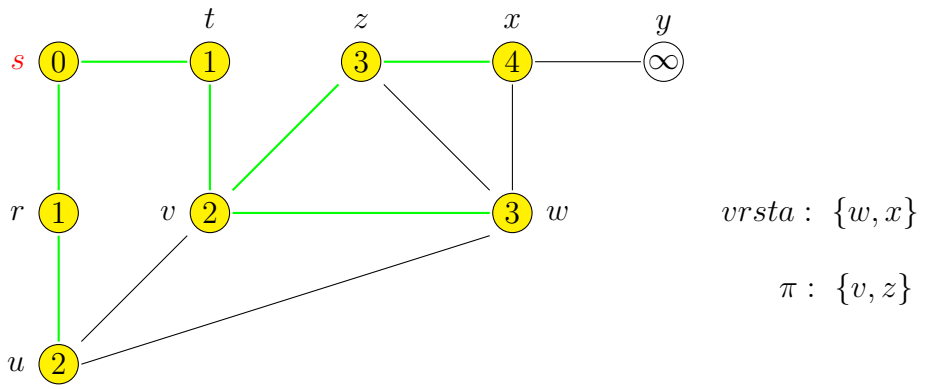
Korak 5. Sosedi vozlišča $u$ so že obiskani. Pogledamo naslednjega kandidata za obisk v seznamu $vrsta$.



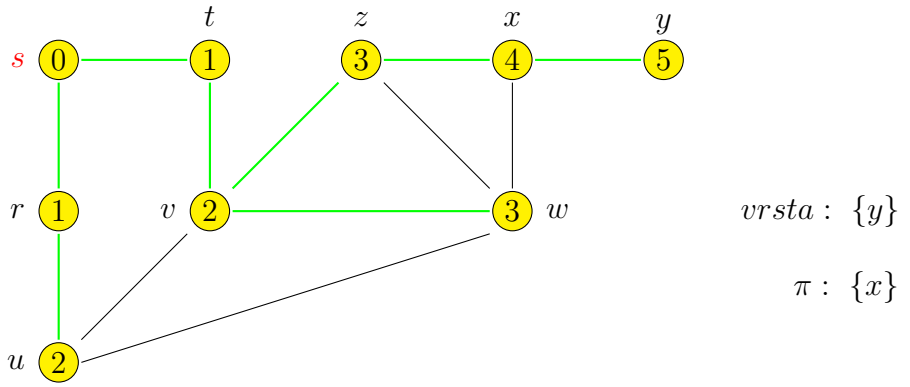

$vrsta: \{z, w\}$

$\pi: \{v, v\}$

Korak 6. Obiščemo vozlišče $x$. Velja $\delta(s,x) = 4$ in $\pi(x) = z$.




Korak 7. Sosedi vozlišča $w$ so že obiskani, zato nadaljujemo z naslednjim kandidatom v seznamu *vrsta*. Obiščemo vozlišče $y$. Velja $\delta(s,y) = 5$ in $\pi(y) = x$.




Ker velja $N(y) = x$, so vsa vozlišča obiskana. Z zelenimi povezavami predstavimo drevo s korenom $s$. Znotraj vozlišč so poračunane dolžine najkrajših poti iz korena $s$ v grafu $G$. Poglejmo si vozlišče $z$, do katerega pridemo po poti $(s,t,v,z)$ ali pa po poti $(s,r,u,w,z)$. Prva pot vsebuje tri povezave, druga pa štiri povezave. Tako lahko vidimo, da je algoritem res vključil najkrajšo pot iz korena do vsakega vozlišča.

♦

# 3. Algoritem

Algoritem *GraphGibbs* smo razvili na podlagi algoritma, ki so ga predstavili Charles Lawrence in njegovi sodelavci [27]. Njihov algoritem je prvi primer uporabe strategije Gibbsovega vzorčenja za reševanje problema iskanja motivov v genetiki. Postavili so dobre temelje za nadaljnji razvoj MCMV metod v bioinformatiki, kar lahko opazimo v velikem številu različic njihovega algoritma, ki smo jih navedli že v uvodnem podpoglavju 1.3. Ker njihov algoritem nima posebnega imena, a je osnova za razvoj algoritma *GraphGibbs*, se bomo v nadaljevanju nanj sklicevali kot na *osnovni algoritem*.

V naslednjem podpoglavju predstavimo *osnovni algoritem* in njegovo psevdokodo, kot ga je razvil Lawrence s sodelavci. Nadaljujemo s predstavitvijo naše ideje za izboljšanje osnovnega algoritma, ki smo jo povzeli iz same genske kode in jo orisali s pomočjo grafične prestavitve sekvenc DNA. Poglavje zaključimo z opisom modifikacij osnovnega algoritma, algoritma *Motif Sampler* [47], in algoritma *Iskanje v širino*, ki so bile potrebne pri implementaciji algoritma *GraphGibbs*.

## 3.1 Osnovni algoritem

Število črk v abecedi $\mathcal{A}$ označimo s črko $K$. Pri osnovnem algoritmu je dana abeceda $\mathcal{A} = \{r_1, \ldots, r_K\}$ lahko bodisi $\mathcal{A}_{DNA}$ bodisi $\mathcal{A}_{Protein}$, kjer

$$\begin{aligned} \mathcal{A}_{DNA} &= \{A, T, C, G\}, & K &= 4 \\ \mathcal{A}_{Protein} &= \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}, & K &= 20. \end{aligned}$$

V nadaljevanju bodo obravnavane DNA sekvence, torej $\mathcal{A} = \mathcal{A}_{DNA}$. Postopek lahko poteka podobno v primeru proteinskih sekvenc.

Vhodna podatka algoritma sta množica $S = \{S_1, S_2, \ldots, S_N\}$ DNA sekvenc in dolžina motiva $W$, ko se motiv pojavi enkrat na sekvenco. Izhod algoritma je končna poravnava $A(v)$ najbolj izrazitega motiva $v$ v dani množici $S$. Dobimo torej vzorčno množico, katere presečni vzorec je najbolj izrazit motiv. Izrazitost motiva se meri s statistiko $I$ (ang. information per parameter), ki maksimira razmerje med frekvencami črk v najdenih "najverjetnejših" vzorcih in frekvencami črk v celi množici brez izbrane vzorčne množice.

Dana množica sekvenc $S$ predstavlja naše podatke $\boldsymbol{y}$. Vektor poravnave $\boldsymbol{a} =$

$(a_1, \ldots, a_N)$ predstavlja vektor neznanih parametrov, ki smo jih v podpoglavju 2.2.1 označili z generično oznako $\boldsymbol{\theta}$. Vrednosti $a_i$, $i = 1, \ldots, N$ predstavljajo položaje začetkov vzorcev v odgovarjajočih sekvencah $S_i$. Ti vzorci pripadajo vzorčni množici, s katero določimo vrednosti tako imenovane pozicijsko uteženе matrike $\mathcal{Q}$ (ang. position weight matrix).

Elementi matrike $\mathcal{Q}$ predstavljajo frekvence pojavitev posamezne črke na vsakem mestu presečnega vzorca in se izračunajo po formuli:

$$q_{i,j} = \frac{c_{i,j} + b_j}{N - 1 + B}, \qquad \text{za } i = 1, \ldots, W \text{ in } \; j = 1, \ldots, K \tag{3.1}$$

kjer $c_{i,j}$ predstavlja dejansko število pojavitev črke $r_j \in \mathcal{A}$ na položaju $i$ v vzorcih, $b_j$ je psevdo število za črko $r_j$ in $B = \sum_j b_j$. Avtorji osnovnega algoritma so vrednost $B$ empirično določili kot $\sqrt{N}$. Psevdo števila, [45], zaznamujejo apriorna pričakovanja pojavitve črk na posameznih položajih motiva. Vsak nukleotid ima pozitivno frekvenco na vseh položajih motiva, saj je $q_{ij}$ različen od nič za vse kombinacije indeksov $i$ in $j$.

Vzporedno razvijemo verjetnostni model frekvenc ozadja (ang. background frequencies) $\mathcal{P} = [p_j]_{j=1,\ldots,K}$ iz trenutne poravnave. Za ozadje štejemo vse dele sekvenc, ki niso bili izbrani v vzorčno množico. Vrednosti $p_j$ tako predstavljajo frekvence črk $r_j \in \mathcal{A}$ v celi množici brez izbranih vzorčnih nizov.

Algoritem najprej slučajno izbere začetne položaje motiva v vseh sekvencah. Drugače povedano, najprej določi slučajni začetni vektor poravnave $\boldsymbol{a}$, tako da z enako verjetnostjo izbere en položaj izmed vseh možnosti v vsaki sekvenci. Nato nastopi iterativni postopek, kjer se izvedeta dva koraka Gibbsovega vzorčevalnika, in sicer napovedni korak (ang. predictive step) in vzorčni korak (ang. sampling step). Število iteracij $M \in \mathbb{N}$ je sicer poljubno, vendar so avtorji predlagali, da je dovolj vzeti $M$ približno enak $10N$.

Pri napovednem koraku algoritem izbere eno izmed sekvenc, $S_z$, slučajno ali glede na določen vrstni red. Indeks $z$ predstavlja komponento v $\boldsymbol{a}$, ki jo bomo na novo vzorčili, glede na vso trenutno poznano informacijo. Trenutno informacijo pa nam predstavljata matrika $\mathcal{Q}$, ki jo dobimo iz poravnave $\boldsymbol{a}_{\setminus z} = \boldsymbol{a} \setminus \{a_z\}$ in vektor $\mathcal{P}$, ki ga določimo na množici $S \setminus \left\{ S_z \cup \left( \boldsymbol{a}_{\setminus z} + W \right) \right\}$. Zapis $\boldsymbol{a}_{\setminus z} + W$ predstavlja vse vzorčne nize, ki se začnejo na mestih $a_i$ v odgovarjajočih sekvencah $S_i$.

V vzorčnem koraku algoritem pregleda vsak segment $x = x_1 x_2 \ldots x_W$ dolžine $W$ v izbrani sekvenci $S_z$ kot možnega kandidata za motiv. Glede na prej dobljeno pozicijsko uteženo matriko $\mathcal{Q}$ in frekvence ozadja se izračunajo verjetnosti $Q_x = P(x|\mathcal{Q})$ in $P_x = P(x|\mathcal{P})$ segmenta $x$.

Verjetnost $Q_x$ izračunamo s pomočjo matrike $\mathcal{Q}$, glede na nukleotide $x_i$, ki so v segmentu $x$, in sicer:

$$Q_x = P(x|\mathcal{Q}) = \prod_{i=1}^{W} q_{i,x_i}.$$

Verjetnost $P_x$ izračunamo analogno z uporabo frekvenc ozadja, glede na nukleotide $x_i$ v segmentu $x$:

$$P_x = P(x|\mathcal{P}) = \prod_{i=1}^{W} p_{x_i}.$$

S temi verjetnostmi utežimo vsakega izmed segmentov $x$:

$$A_x = \frac{Q_x}{P_x} = \frac{P(x|\mathcal{Q})}{P(x|\mathcal{P})}.$$

Novi kandidat za motiv določimo med uteženimi segmenti, tako da enega slučajno izberemo z verjetnostjo $A_x / \sum_s A_s$, kjer vsota teče po vseh možnih segmentih. Položaj $p$ novega segmenta $x^p$ v sekvenci $S_z$ predstavlja vrednost $a_z$ v vektorju $\boldsymbol{a}$.

Posamezna poravnava se oceni s pomočjo vrednosti:

$$F = \sum_{i=1}^{W} \sum_{j=1}^{K} c_{i,j} \log \frac{q_{i,j}}{p_j}, \tag{3.2}$$

kjer sta vrednosti $c_{i,j}$ in $q_{i,j}$ izračunani iz kompletne poravnave, torej je vključena tudi sekvenca $S_z$. Kot v formuli 3.1, vrednost $c_{i,j}$ predstavlja dejansko število pojavitev črke $r_j \in \mathcal{A}$ na položaju $i$ v vseh vzorcih.

Ker algoritem išče rešitev lokalno, obstaja možnost, da se ujame v lokalni ekstrem. Temu se lahko izognemo, tako da pri vsaki $m$-ti iteraciji (za nek poljuben $m \leq M$) primerjamo trenutno poravnavo $\boldsymbol{a}$ s poravnavami, ki so zamaknjene levo in desno od trenutne. Velikost zamika je sorazmeren z dolžino $W$. Za vsako zamaknjeno poravnavo se izračuna vrednost $F$ in primerja z vrednostjo $F$ trenutne poravnave.

Ocena $F$ zadostuje v primeru, kadar določimo samo eno možno dolžino $W$ motiva, medtem ko je dejanska dolžina motiva uporabniku neznana. Potrebno je torej pogledati vzorčne množice za različne vrednosti $W$ kot možne rešitve. Avtorji [27], so obravnavali možnost, kjer uporabnik določi območje možnih dolžin motiva in algoritem določi optimalno dolžino. Vendar v tem primeru ocena $F$ ni več primeren kriterij, saj njena vrednost narašča z večanjem dolžine motiva $W$. Avtorji so zato predlagali drug kriterij, ki temelji na razmerju nepopolnih podatkov in logaritmične verjetnosti (ang. incomplete-data log-probability):

$$G = F - \sum_{z=1}^{N} \left( \log L'_z + \sum_{p=1}^{L'_z} Y_{z,p} \log Y_{z,p} \right),$$

kjer je $L'_z$ število vseh možnih položajev za vzorec znotraj sekvence $S_z$, $Y_{z,p}$ pa je normalizirana utež segmenta na položaju $p$ oziroma predstavlja kvocient med utežjo $A_{x^p} = \frac{Q_{x^p}}{P_{x^p}}$ in vsoto uteži vseh segmentov znotraj sekvence $S_z$. S pomočjo vrednosti $G$ lahko izračunamo statistiko $I$:

$$I = \frac{G}{(K-1)W}, \tag{3.3}$$

Psevdokoda osnovnega algoritma je podana v algoritmu 4.

**Algoritem 4** Psevdokoda *osnovnega algoritma.*

**Vhod:** Množica $S = \{S_1, \ldots, S_N\}$ sekvenc DNA in širina motiva $W$.
**Izhod:** Motiv širine $W$ z odgovarjajočo poravnavo $\boldsymbol{a}$.

1: Vzemi poljubno začetno poravnavo $\boldsymbol{a} = \{a_1, \ldots, a_N\}$.
2: **repeat**
3: Izberi sekvenco $S_z$ in jo izloči iz $S$, skupaj z njenim položajem $a_z$ v $\boldsymbol{a}$.
4: Izračunaj $\mathcal{Q}$ in $\mathcal{P}$ iz $S \backslash \{S_z\}$.
5: Uteži vse segmente $x^p$, $p = 1, \ldots, L'_z$, v sekvenci $S_z$ z utežmi $A_{x^p}$.
6: Določi segment $x^p$ slučajno in določi novo vrednost $a_z = p$.
7: **until** ni dosežena konvergenca ali $M$

**vrni** poravnavo $\boldsymbol{a}$ motiva

## 3.2 Modeliranje sekvenc DNA z grafom

Večina različic osnovnega algoritma se osredotoči na izboljšavo kriterijske funkcije $F$ oziroma $G$, kar zato pomeni spremembo statistike $I$. S slednjo spremembo se kontrolirajo premiki po prostoru rešitev, to je v katero smer se premika Markovska veriga. Mi smo pogledali na možnost izboljšanja algoritma z drugega vidika, in sicer kje začeti z Gibbsovim vzorčenjem. Z dobro oceno začetne točke imamo večjo verjetnost, da pridemo do globalnega ekstrema, ki zaznamuje iskano stacionarno ciljno porazdelitev $\pi(\boldsymbol{\theta})$.

Gibbsovo vzorčenje išče rešitev lokalno, zato začetna točka iterativnega postopka ni tako trivialna, saj vpliva na hitrost konvergence vzorčevalnika in usmeritev proti enemu od lokalnih ekstremov. Lokalno iskanje je praviloma hitrejše, vendar se lahko algoritem ujame v lokalni optimum ali plato. Po eni strani se temu izognemo tako, da v toku iterativnega postopka periodično primerjamo trenutno točko v prostoru rešitev s točkami v njeni okolici glede na izbrano statistiko. Po drugi strani pa lahko povečamo verjetnost, da algoritem najde globalni ekstrem, z dobro izbiro začetne točke Gibbsovega vzorčevalnika. V tem primeru začetna točka ni izbrana slučajno, ampak je delno določena. Oba načina sta prisotna v našem algoritmu, kjer delno določimo začetno točko in nato periodično preverjamo smer Gibbsovega vzorčenja.

Za izbiro dobre začetne točke predlagamo naslednjo metodo. Vzamemo dane vhodne sekvence in jih združimo v eno sekvenco $R$. Ker DNA zapis uporabi trojčke kot osnovne elemente (genski kod), smo zakodirali DNA sekvence v trojčke in uporabili Gibbsovo vzorčenje na kodiranih sekvencah. Branje trojčkov v originalnih sekvencah poteka z leve proti desni s premikom za en nukleotid . Na primer, niz AATCGTGGC se kodira kot AAT, ATC, TCG, CGT, GTG, TGG, GGC. Ker ima abeceda za DNA sekvence štiri črke, je število vseh možnih trojčkov ravno $4^3 = 64$. Vsakega od trojčkov označimo s številko, kot smo jih v tabeli 3.1.

Sekvenco $R$ zakodiramo po zgornjem postopku in dobimo kodirano sekvenco $R_c$. Abeceda $\mathcal{A}_{kod}$ sekvence $R_c$ je sestavljena iz števil $1, 2, \ldots, 64$, ki označujejo posa-

Tabela 3.1: Številčne oznake trojčkov.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | AAA | 9 | ACA | 17 | TAA | 25 | TCA | 33 | CAA | 41 | CCA | 49 | GAA | 57 | GCA |
| 2 | AAT | 10 | ACT | 18 | TAT | 26 | TCT | 34 | CAT | 42 | CCT | 50 | GAT | 58 | GCT |
| 3 | AAC | 11 | ACC | 19 | TAC | 27 | TCC | 35 | CAC | 43 | CCC | 51 | GAC | 59 | GCC |
| 4 | AAG | 12 | ACG | 20 | TAG | 28 | TCG | 36 | CAG | 44 | CCG | 52 | GAG | 60 | GCG |
| 5 | ATA | 13 | AGA | 21 | TTA | 29 | TGA | 37 | CTA | 45 | CGA | 53 | GTA | 61 | GGA |
| 6 | ATT | 14 | AGT | 22 | TTT | 30 | TGT | 38 | CTT | 46 | CGT | 54 | GTT | 62 | GGT |
| 7 | ATC | 15 | AGC | 23 | TTC | 31 | TGC | 39 | CTC | 47 | CGC | 55 | GTC | 63 | GGC |
| 8 | ATG | 16 | AGG | 24 | TTG | 32 | TGG | 40 | CTG | 48 | CGG | 56 | GTG | 64 | GGG |

mezno kombinacijo črk abecede $\mathcal{A}_{DNA}$ kot v tabeli 3.1. Par števil z razmikom dva v sekvenci $R_c$ je enakovreden paru trojčkov v sekvenci $R$. Kadar dolžina sekvence $R$ ni večkratnik števila tri, potem pri branju zaporednih trojčkov v $R$ lahko izpustimo zadnji dve črki ali samo zadnjo črko. Temu se izognemo, tako da preberemo sekvenco $R_c$ trikrat z razmikom dva, to je preberemo vsako tretje število. Preden branje sekvence $R_c$ ponovimo, izpustimo prvo število (pri drugem branju) oziroma prvi dve števili (pri tretjem branju). S tem zajamemo povezave vseh možnih zaporednih trojčkov znotraj sekvence $R$. Za zgled si poglejmo kodiranje in branje naslednje sekvence $R$ dolžine $L = 20$:

$$\begin{aligned}
\text{Sekvenca } R &: \text{CTAGGCCCAATATCTTGATA} \\
\text{Sekvenca } R_c - \mathcal{A}_{DNA} &: \text{CTA, TAG, AGG, GGC, GCC, CCC, CCA, CAA, AAT,} \\
&\quad \text{ATA, TAT, ATC, TCT, CTT, TTG, TGA, GAT, ATA} \\
\text{Sekvenca } R_c - \mathcal{A}_{kod} &: 37, 20, 16, 63, 59, 43, 41, 33, 2, 5, 18, 7, 26, 38, 24, 29, 50, 5
\end{aligned}$$

Prvo branje.

$R_c : 37, 63, 41, 5, 26, 29$

$\cong R :$ CTA, GGC, CCA, ATA, TCT, TGA TA

Drugo branje.

$R_c : 20, 59, 33, 18, 38, 50$

$\cong R :$ C TAG, GCC, CAA, TAT, CTT, GAT A

Tretje branje.

$R_c : 16, 43, 2, 7, 24, 5$

$\cong R :$ CT AGG, CCC, AAT, ATC, TTG, ATA

Med branjem sproti gradimo matriko povezav, ki je kvadratna matrika $64 \times 64$, katere elementi so frekvence pojavitev parov zaporednih trojčkov v sekvenci $R$. Če ovrednotimo pozitivne elemente v matriki povezav z 1, dobimo matriko sosednosti usmerjenega multigrafa $G$ reda 64. Vozlišča v $G$ predstavljajo trojčke in so označena

s številkami $1, 2, \ldots, 64$; ena številka označuje trojček. Povezava med dvema vozliščema v $G$ zrcali sosednost med dvema zaporednima trojčkoma v $R$. Tako se število povezav med dvema vozliščema ujema s številom odgovarjajočih trojčkov sosednih v sekvenci $R$. Povezave so usmerjene, saj se sekvenca $R$ bere z leve proti desni in je zaporedje trojčkov pomembno.

Ko končamo z branjem kodirane sekvence $R_c$, dobljeno matriko povezav normaliziramo po vrsticah (posamezni element v vrstici delimo z vsoto vseh elementov vrstice) in nato poiščemo najvišjo vrednost posamezne vrstice preko vseh stolpcev, ki se lahko pojavi tudi v več stolpcih. Na ta način utežimo povezavo med dvema vozliščema. Potem sestavimo novo matriko sosednosti, tako da s številko 1 zamenjamo najvišjo vrednost po posamezni vrstici v normalizirani matriki povezav, torej najvišjo utež, ostale vrednosti pa postavimo na 0.

Vozlišča uredimo v nenaraščajoče zaporedje glede na njihovo najbolj uteženo izhodno povezavo in zberemo prvih deset vozlišč v množico $\mathcal{V} = \{v_1, v_2, \ldots, v_{10}\}$. Število izbranih vozlišč je sicer poljubno, vendar smo preko praktičnih poskusov določili, da se razlika med utežmi opazno poveča okoli desetega vozlišča. Izbrana vozlišča, skupaj s svojim sosedom predstavljajo besede dolžine 6, ki se najpogosteje pojavijo v $R$. Ker nas zanimajo samo visoke pojavitve posameznih nizov znotraj $R$, se omejimo samo na ta vozlišča.

Naj bo $\mathcal{N} = \cup\{N^-(u)\}_{u\in\mathcal{V}}$, kjer je $N^-(u)$ množica vseh sosednjih krajišč v izhodnih lokih vozlišča $u \in \mathcal{V}$. Zanima nas induciran podgraf $H = G[\mathcal{N}]$. Vsako vozlišče $v_s \in \mathcal{V}$ je lahko izvorno vozlišče za sprehod dolžine $\ell = \lceil W/3 \rceil - 1$. Dolžina sprehoda je določena s številom trojčkov v zapisu motiva dolžine $W$. Kadar je dolžina $W$ večkratnik števila tri, potem je motiv sestavljen iz $W/3$ trojčkov. V podgrafu $H$ obiščemo torej $W/3$ paroma povezanih ogljišč, ko se sprehodimo po $W/3-1$ povezavah oziroma lokih. V primeru, ko velja $W \not\equiv 0 (\mathrm{mod}\ 3)$, vzamemo naslednji večji večkratnik števila tri, to je določimo $W + 3 > W' > W$ in $W' \equiv 0 (\mathrm{mod}\ 3)$. Znotraj daljšega motiva je namreč krajši motiv in tako ne izgubimo nobene črke v nizu motiva. Vsak sprehod zrcali zaporedje trojčkov v sekvenci $R$ in je naravni način modeliranja nadpovprečno zastopanih vzorcev v $R$. Podgraf $H$ lahko vsebuje tudi zanke, zato tukaj obravnavamo sprehode po grafu in ne poti.

Pregledamo vse sprehode izračunane dolžine s prilagojenim algoritmom *Iskanje v širino*. Njegova osnovna različica je opisana z algoritmom 3. Algoritem smo prilagodili tako, da so vozlišča lahko obiskana več kot enkrat. S tem upoštevamo tudi zanke v grafu $H$. V našem primeru imamo določeno dolžino sprehoda $\ell$ in nas zanima, koliko je sprehodov te dolžine iz vsakega od vozlišč iz množice $\mathcal{V}$. Te sprehode shranimo v seznamu *sprehodi*. Ker želimo najti povezavo z najvišjo frekvenco obiska, hkrati preštejemo še število obiskov posameznega vozlišča. Večkrat kot je bilo vozlišče v teku algoritma obiskano, na več različnih sprehodih se je le-to pojavilo. To naprej nakazuje, da sta odgovarjajoče vozlišče in njegov sosed v največjem od presekov vseh najdenih sprehodov. Število obiskov spremljamo s seznamom *stopnja* dolžine 64.

Nas zanimajo sprehodi, zato tukaj ne vodimo seznama očetov, kot smo ga vodili v algoritmu 3. Dodatno smo spremenili seznam logičnih vrednosti *obiskani* v seznam števil dolžine 64, kajti merimo, kolikokrat je bilo določeno vozlišče obiskano. Vrsta *korak* ima podobno vlogo kot vrsta *vrsta* v originalnem algoritmu. V algoritmu 5 je zapisana psevdokoda prilagojenega algoritma *Iskanje v širino*.

**Algoritem 5** Psevdokoda prilagojenega algoritma *Iskanje v širino*.

**Vhod:** Seznam sosedov $N(u)$ za vse $u \in V(H)$, množica $\mathcal{V}$ in dolžina sprehoda $\ell$.
**Izhod:** Seznam sprehodov *sprehod* in seznam vseh obiskov *stopnja*.

1: Ustvari seznama *sprehod* in *stopnja*. Nastavi *korakIndeks* na 0.
2: **for all** vozlišče $v \in \mathcal{V}$ **do**
3: Ustvari seznam *obiskani* in vrsto *korak*. Nastavi *steviloKopij* na 0.
4: $obiskani(v)$ += 1
5: $stopnja(v)$ += 1
6: $steviloKopij = 1$
7: Vstavi $(korak, v)$.
8: Vstavi $(sprehod, korak)$.
9: **for** $i = 1 : \ell$ **do**
10: $steviloSprehodov = steviloKopij$
11: **for** $k = 1 : steviloSprehodov$ **do**
12: $korak = sprehod(korakIndeks + k)$
13: vozlišče $u = korak(i)$
14: $steviloKopij$ += $|N(u)| - 1$
15: **for all** $w \in N(u)$ **do**
16: $obiskani(w)$ += 1
17: **if** $obiskani(w) < 3$ **then**
18: $stopnja(w)$ += 1
19: **end if**
20: Vstavi $(korak, w)$.
21: **if** $w$ prvi sosed **then**
22: Vstavi $(sprehod, korak)$.
23: **else**
24: $sprehod(korakIndeks + k) = korak$
25: **end if**
26: **end for**
27: **end for**
28: **end for**
29: **end for**

**vrni** seznama *sprehod* in *stopnja*.

Pri algoritmu *Iskanje v širino* poiščemo vse nakrajše poti iz korena $v_s$ do vsakega dosegljivega vozlišča v grafu, izrek 2.4.1. Pri prilagojenem algoritmu 5 iščemo vse sprehode določene dolžine $\ell$ iz korena $v_s \in \mathcal{V}$. Koren $v_s$ je izmenično vsako od vozlišč v množici $\mathcal{V}$, iz katerega iščemo vse sprehode dolžine $\ell$. Ker dopustimo možnost, da

ima graf $H$ zanke, lahko iščemo samo sprehode in ne poti. Zanka v grafu $H$ lahko ustvari cikel v postopku, zato sledimo številu obiskov posameznega vozlišča, ki je lahko največ dve, s seznamom *obiskani*. Z vrsto *stopnja* beležimo število obiskov vozlišča globalno, torej pri pregledu vseh možnih sprehodov iz vseh možnih korenov.

Iz vrste *stopnja* ustvarimo množico vseh različnih vrednosti v vrsti *stopnja*, ki jo označimo z $\Delta$. Vozlišča v $H$ grupiramo v množice $\mathcal{K}_{\delta_i}$ glede na vrednosti $\delta_i \in \Delta$, kjer je $i \in \{1, 2, \ldots, |\Delta|\}$. Vozlišča z enakim številom obiskov $\delta_i$ spadajo v isto skupino $\mathcal{K}_{\delta_i}$. Vsako vozlišče iz $\mathcal{K}_{\delta_i}$ skupaj z njegovim sosedom vzamemo kot del vzorca in predstavlja *kandidata* za končen motiv. Dve vozlišči predstavljata dva trojčka, torej ima vzorec začetno dolžino enako 6. Naj bo $\psi(k)$ število pojavitev vzorca $k$ v množici sekvenc $S$ in $E$ število pričakovanih pojavitev motiva. Slednje število lahko določi uporabnik ali pa se vzame privzeto vrednost $E = N$, saj predpostavljamo, da se v vsaki sekvenci motiv pojavi vsaj enkrat.

Naj bo $\delta = \max_{\delta_i \in \Delta}\{\delta_i\}$. Označimo množico vozlišč s številom obiskov $\delta$ kot $\mathcal{K}_\delta$ in rešitev spodnjega postopka s $k_R$. Na začetku sta dolžina motiva $k_R$ in število pojavitev enaka nič. Dolžina iskanega motiva $W$ je lahko določena s strani uporabnika (možnost $a$), drugače pa algoritem *GraphGibbs* v prvem delu poišče najdaljši popolnoma ohranjen motiv (možnost $b$) s številom pojavitev $\psi(k_R) > \lfloor E/2 \rfloor$ z naslednjim postopkom v šestih korakih:

1. Vzemimo kandidata $k \in \mathcal{K}_\delta$. Označimo dolžino kandidata $k$ z $\ell_k$ in mejo za število pojavitev motiva z $B = \lfloor E/2 \rfloor$.

2. Preštejemo število pojavitev $\psi(k)$ kandidata $k$ v vhodnih sekvencah. Če je število $\psi(k) \geq B$ in $W = 6$ (možnost $a$), potem pri pogoju $\psi(k_R) < \psi(k)$ nastavimo $k_R = k$ in se premaknemo na peti korak. Kadar je $W > 6$ (možnost $a$) ali pa $W$ ni določen (možnost $b$) in velja $\psi(k) \geq B$, nadaljujemo s tretjim korakom. V primeru, ko je $\psi(k) < B$ pa nadaljujemo z začetnim krajiščem kandidata $k$ v tretjem koraku.

3. Vzamemo črko $r \in \mathcal{A}_{DNA}$ in preštejemo število pojavitev vzorca $k^* = r{+}k$, kjer operator $+$ predstavlja spojitev črke $r$ in kandidata $k$ v danem zaporedju. Če je $\psi(k^*) \geq B$ postavimo $k = k^*$ in ponovimo tretji korak. Korak ponavljamo, dokler ni $\ell_{k^*} = W$ (možnost $a$) ali pa $\psi(k^*) < B$ za vse $r \in \mathcal{A}_{DNA}$. V prvem primeru določimo $k_R = k^*$, če velja $\psi(k_R) < \psi(k^*)$, in nadaljujemo s petim korakom, v drugem pa nadaljujemo z naslednjim korakom.

4. Vzamemo črko $r \in \mathcal{A}_{DNA}$ in pogledamo število pojavitev vzorca $k' = k^* + r$. V primeru, da je $\psi(k') \geq B$ nastavimo $k^* = k'$ in ponovimo četrti korak, dokler ne velja $\ell_{k'} = W$ (možnost $a$) ali pa je $\psi(k') < B$ za vse možne vzorce $k'$. Kadar je $\psi(k_R) < \psi(k')$ nastavimo $k_R = k'$ in nadaljujemo z naslednjim korakom.

5. Če $E$ ni določen, potem nastavimo $E = \max(\lfloor \psi(k')/N \rceil, N)$. Izločimo kandidata $k$ iz $\mathcal{K}_\delta$. Dokler je $\mathcal{K}_\delta$ neprazna, se vrnemo na prvi korak. Drugače pa izločimo vrednost $\delta$ iz množice $\Delta$, torej $\Delta = \Delta \backslash \{\delta\}$. Za vrednost $\delta$ vzamemo nov največji element v množici $\Delta$. Če je $\delta \neq 0$ se vrnemo na prvi korak,

sicer pa nadaljujemo s šestim korakom. Kadar velja $\delta = 0$ in $M = 1$, za $k_R$ izberemo najdaljši vzorec z najvišjim številom pojavitev med vsemi kandidati $k^*$ iz tretjega koraka in $k'$ iz četrtega koraka preden postopek iskanja motivov zaključimo.

6. Kandidat $k_R$ ustreza danim parametrom, to je: (možnost $a$) $\psi(k_R) \geq B$ in $\ell_{k_R} = W$ ali (možnost $b$) $\psi(k_R) \geq B$ in $\ell_{k_R} > \ell_{k_j}$ za vse možne kandidate $k_j$. Poravnavo motiva $A(k_R)$ vzamemo iz dane množice sekvenc $S$. Kadar je število različnih motivov $M$ določeno in velja $M = 1$, postopek končamo, drugače pa ponovno postavimo model grafa, kot je opisano spodaj, in ponovimo celoten postopek na novem grafu.

Rešitev je zadnji najdaljši zapis $k_R$ oziroma motiv dolžine $W$ (možnost $a$), ki ima v množici sekvenc $S$ najvišje število pojavitev. Torej daljši motiv z manjšim številom pojavitev upoštevamo za nadaljnjo analizo tudi zato, ker so krajši motivi del daljšega in jih tako ne zavrnemo. Ko je motiv zgrajen, shranimo vse pojavitve tega motiva v vektor oziroma matriko poravnave.

Poravnavo motiva $A(k_R)$ si zapomnimo in izbrišemo vse vzorčne nize $\{a_i+W\}_{a_i \in A(k_R)}$ (možnost $a$) oziroma $\{a_i + \ell_{k_R}\}_{a_i \in A(k_R)}$ (možnost $b$) iz množice sekvenc $S$. Postopek prvega dela algoritma ponavljamo na skrajšanih sekvencah, dokler ne najdemo vseh možnih vzorcev ali do danega števila motivov. Z izločitvijo vzorčnih nizov se pri vsaki ponovitvi matrika sosednosti spremeni in s tem tudi odgovarjajoči graf. Vzorec, ki ga algoritem najde na novem grafu, ima lahko manjše število pojavitev v začetni množici $S$. V tem primeru z algoritmom poiščemo krajši podniz znotraj vzorca, ki ima večje število pojavitev znotraj množice $S$. Dolžina podniza $W'$ je odvisna od dolžine $W$ oziroma $\ell_{k_R}$, tako da je razlika $1 \leq W - W' \leq 6$. Na primer: za $W = 6$ je $W' = 5$; za $W = 14$ je $W' = 10$; za $W = 30$ je $W' = 24$.

Kadar pri zgornjem postopku ne najdemo nobenega motiva pri prvem iskanju, algoritem vrne izjemo *V množici ni nobenega motiva.* S tem povemo, da ni nobenega popolno ohranjenega motiva, ki se statistično nadpovprečno pojavi v množici $S$. V tem primeru vzamemo kandidata z najvišjo vrednostjo $\psi(\cdot)$ in njegovo poravnavo za nadaljnjo obravnavo z Gibbsovim vzorčevalnikom.

Vsak od popolnoma ohranjenih motivov $y_i$, $i = 1, \ldots, m$, ki jih najdemo v prvem delu, predstavlja del rešitve, saj je to vzorec, ki se v množici $S$ pojavi dovolj pogosto. Kjer velja $\psi(y_i) < E$, s poravnavo motiva $y_i$ delno določimo začetno točko Gibbsovega vzorčevalnika. Manjkajoče vrednosti izberemo slučajno. Z Gibbsovim vzorčevalnikom poiščemo še variacije motiva $y_i$. Z delno določeno točko usmerimo vzorčevalnik proti iskani končni rešitvi, torej vzorčni množici relativno ohranjenega motiva.

Določene izjeme in omejitve pri nadaljevanju algoritma, ki so posledica prvega dela, obravnavamo posebej v poglavju 6. S primerom 3.2.1 je dodatno ponazorjen postopek prvega dela na konkretnem primeru. Celoten algoritem *GraphGibbs* pa je orisan v psevdokodi 7.

**Primer 3.2.1.** Za ponazoritev postopka prvega dela algoritma *GraphGibbs* si poglejmo enostaven primer, kjer iščemo popolnoma ohranjen motiv dolžine $W = 9$, ki se v petih sekvencah dolžine $L_i = 109$ za vsak $i$ pojavi natanko enkrat. Torej v celi množici DNA sekvenc $S = \{S_1, \ldots, S_5\}$ je pričakovano število pojavitev enako $E = 5$. Iščemo motiv:

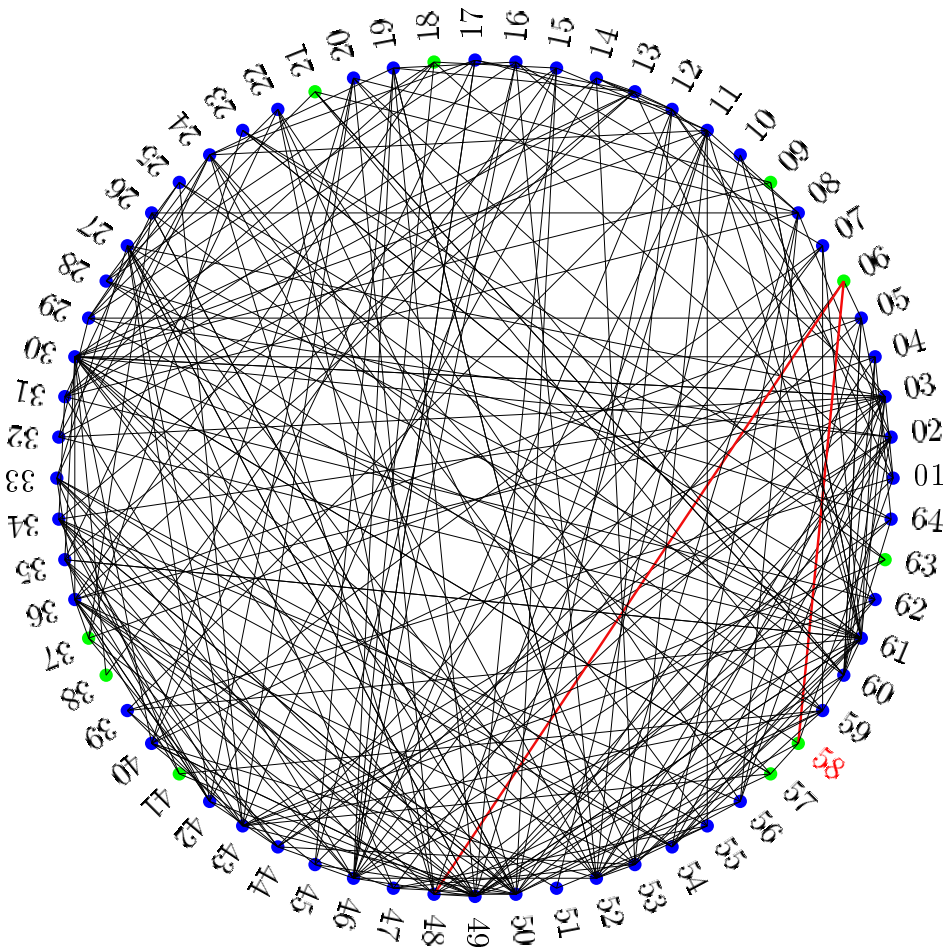
$$y = \text{GCTATTCGG}.$$

Izpis algoritma *GraphGibbs* za ta primer je podan v prilogi A.

Najprej zberemo sekvence $S_i$ v eno dolgo sekvenco $R = S_1 + \ldots + S_5$, kjer operator $+$ predstavlja spojitev dveh sekvenc, kot v primeru 1.2.12. Potlej sekvenco $R$ zakodiramo s številkami glede na tabelo 3.1. Naš motiv $y$ je v kodi enak:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\underline{GCT}$ | $\longrightarrow$ | $CTA$ | $\longrightarrow$ | $TAT$ | $\longrightarrow$ | $\underline{ATT}$ | $\longrightarrow$ | $TTC$ | $\longrightarrow$ | $TCG$ | $\longrightarrow$ | $\underline{CGG}$ |
| 58 | $\longrightarrow$ | 37 | $\longrightarrow$ | 18 | $\longrightarrow$ | 6 | $\longrightarrow$ | 23 | $\longrightarrow$ | 28 | $\longrightarrow$ | 48 |

Pri branju z leve proti desni naredimo $64 \times 64$ matriko povezav $P$. Matriko $P$

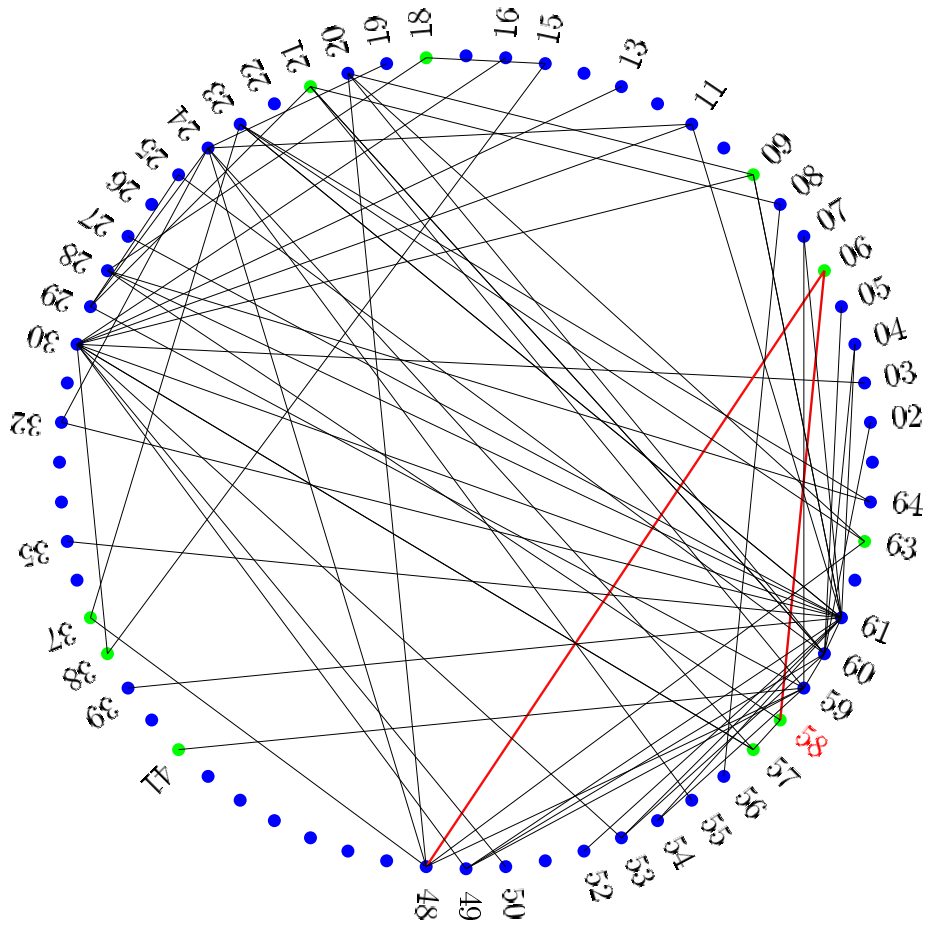


Slika 3.1: Skica grafa $G$. Z zeleno barvo so označena vozlišča $\mathcal{V}$, z rdečo barvo pa je označen sprehod za motiv $y$.

normaliziramo po vrsticah in s tem utežimo posamezno povezavo med zaporednima trojčkoma v sekvenci $R$. Glede na najvišje uteži v vsaki od vrstic matrike $P$ oblikujemo matriko sosednosti $T$ in določimo 10 vrstic (vozlišč) z najvišjimi vrednostmi, od koder dobimo:

$$\begin{aligned}\mathcal{V} &= \{58, 37, 38, 6, 18, 21, 63, 57, 9, 41\} \\ &= \{GCT, CTA, CTT, ATT, TAT, TTA, GGC, GCA, ACA, CCA\}.\end{aligned}$$

Slika 3.1 je skica grafa $G = (V, T)$, kjer so v množici $V$ zbrani vsi trojčki. Graf $G$ je multigraf, saj lahko vsebuje tudi zanke. Graf $G$ je sicer usmerjen, vendar so na sliki 3.1 povezave prikazane neusmerjeno zaradi večje preglednosti. Prav tako smo vozlišča označili z odgovarjajočimi številkami in jih pobarvali modro. Zelena barva predstavlja vozlišče iz množice $\mathcal{V}$. Z rdečo barvo pa je označen sprehod dolžine $\ell = \lceil W/3 \rceil - 1 = 2$ za motiv $y$, ki se začne v vozlišču z rdečo oznako.



Slika 3.2: Skica induciranega podgrafa H. Z zeleno barvo so označena vozlišča iz $\mathcal{V}$ in z rdečo barvo je označen sprehod za motiv $y$.

Definirajmo $\mathcal{N} = \cup\{N^-(u)\}_{u\in\mathcal{V}}$, kjer je $N^-(u)$ množica sosednjih krajišč izhodnih povezav vozlišča $u \in \mathcal{V}$. Okolica vozlišča 58, na primer, je $N(58) = \{1, 6, 28, 49, 57\}$, pri čemer je $N^-(58) = \{6\}$ in $N^+(58) = \{1, 28, 49, 57\}$. Za naš primer dobimo

$$\mathcal{N} = V - \{1, 10, 12, 14, 17, 31, 33, 34, 36, 40, 42, 43, 44, 45, 46, 47, 51, 62\}$$

velikosti $|\mathcal{N}| = 46$. Definirajmo induciran podgraf $H = G[\mathcal{N}]$, katerega skica se nahaja v sliki 3.2. Tako kot prej so vsa vozlišča v $V$ označena z modro barvo razen vozlišč iz $\mathcal{V}$, ki so zelene barve. Sprehod, po katerem izpeljemo iskan motiv $y$ je narisan rdeče z začetkom v vozlišču 58. Okolica vozlišča 58 v podgrafu $H$ je sedaj $N_H(58) = \{6, 28, 57\}$, kjer je $N_H^-(58) = \{6\}$ in $N_H^+(58) = \{28, 57\}$.

Na grafu $H$ poiščemo vse sprehode dolžine $\ell$ z začetnimi vozlišči iz $\mathcal{V}$ s pomočjo prilagojenega algoritma *Iskanja v širino*. V tem primeru dobimo 70 različnih sprehodov in množico $\Delta = \{5, 4, 3, 2, 1, 0\}$. Definiramo $\delta = 5$. V množici kandidatov $\mathcal{K}_\delta$ dobimo 17 možnosti. Gradnja motiva do dane dolžine $W = 9$ je prikazana v izpisu algoritma *GraphGibbs* v prilogi A. Izpisane so samo možnosti, ki imajo vsaj eno pojavitev v celi množici $S$. V stolpcu `našli` je zapisano število pojavitev glede na pričakovano število pojavitev v celi množici $S$, v stolpcu `našli` % pa je podan odstotek najdenih pričakovanih pojavitev.

Ker je motiv $y$ v tem primeru popolnoma ohranjen, so bile vse pojavitve motiva najdene že v prvem delu. Drugače se požene Gibbsov vzorčevalnik, s katerim algoritem poišče vse pričakovane položaje motiva. V danem primeru smo iskali samo en motiv v množici $S$, zato postopek iskanja končamo. Če bi želeli najti več motivov, potem bi odstranili vse vzorčne nize motiva $y$ iz množice $S$ in ponovili postopek na preostalem delu množice $S$.

♦

## 3.3 Modifikacija osnovnega algoritma

Pri osnovnem algoritmu se začetna točka iteracije določi slučajno, tako da slučajno določimo vrednosti vektorja poravnave $\boldsymbol{a}$. Dodatna predpostavka je, da se motiv pojavi v vsaki sekvenci. Kadar pričakujemo, da se motiv pojavi večkrat na sekvenco, se vektor poravnave spremeni v matriko $A$. Pri osnovnem algoritmu predpostavimo, da je število pričakovanih motivov enakomerno porazdeljeno po sekvencah. Če se motiv pojavi $e$-krat v vsaki sekvenci $S_i \in S$, je poravnava motiva izražena v matriki dimenzij $N \times e$. Pri osnovnem algoritmu tako na začetku določimo $E = N \cdot e$ slučajnih vrednosti za matriko $A$.

Za razliko od *osnovnega algoritma* 4, je pri algoritmu *GraphGibbs* začetna točka, to je začetna poravnava $\boldsymbol{a}^0$ deloma določena. S tem Gibbsovemu vzorčevalniku povemo, kje naj začne iskati. Še vedno obdržimo element slučajnosti Gibbsove metode, vendar smo s prvim delom dobili dodatne informacije. Pri osnovni metodi poznamo samo sekvence v množici $S$, z *GraphGibbs* metodo pa dobimo še nekaj informacij o njihovi notranji strukturi, namreč dodatno izvemo, kateri popolnoma ohranjeni vzorci se dovolj pogosto pojavijo skupaj z njihovimi poravnavami.

Recimo, da smo v prvem delu našli motiv $y$ dolžine $w$ s poravnavo $A(y)$. Pri predpostavki, da je število pričakovanih pojavitev enako v vsaki sekvenci, je poravnava $A(y)$ matrika, ki ima po prvem delu določenih $\psi(y)$ elementov, kjer s $\psi(y)$ označimo število pojavitev motiva $y$ v množici $S$.

Pri osnovni metodi je potrebno definirati vse elemente poravnave, zato moramo slučajno določiti še $E - \psi(y)$ vrednosti v poravnavi $A(y)$. Nato nadaljujemo s postopkom osnovnega algoritma. Predpostavka, da je število pričakovanih pojavitev motiva enako v vsaki sekvenci $S_i$, omogoča lažje delovanje osnovnega algoritma, vendar redko drži pri realnih množicah podatkov. Na to smo med drugim opozorili že pri opisu raziskovalnega problema v podpoglavju 1.2. Predpostavko, da so motivi neenakomerno porazdeljeni po sekvencah v podani množici $S$, je torej potrebno vključiti v model.

Pri razvoju algoritma *GraphGibbs* smo slednjo predpostavko vključili s pomočjo izračuna porazdelitvene funkcije $\Gamma$, po kateri so motivi porazdeljeni po sekvencah. S pomočjo funkcije $\Gamma$ modeliramo neenakomerno porazdeljene motive v dani množici sekvenc. Idejo smo povzeli iz algoritma *Motif Sampler* [47]. V svojem članku so avtorji predstavili način, s katerim določimo število pričakovanih pojavitev motiva na sekvenco.

Za boljše razumevanje implementacije porazdelitvene funkcije $\Gamma$ v naš algoritem si najprej na kratko oglejmo način določitve porazdelitve $\Gamma$ v algoritmu *Motif Sampler*. Psevdokodo postopka najdemo v algoritmu 6. Tako kot pri *osnovnem algoritmu* 4, tudi pri algoritmu *Motif Sampler* poznamo samo množico DNA sekvenc $S$ in pričakovano dolžino motiva $W$.

Postopek začnemo z definiranjem nove slučajne spremenljivke $E_i$, ki meri število pojavitev motiva $y$ v sekvenci $S_i$. Verjetnost, da imamo $c$ pojavitev motiva v sekvenci $S_i$, označimo kot:

$$\gamma_i(c) = P(E_i = c|S_i, \mathcal{Q}, \mathcal{P}), \tag{3.4}$$

kjer so elementi matrike $\mathcal{Q} = [q_{ij}]_{\substack{i=1,\ldots,W \\ j=1,\ldots,K}}$ izračunani po formuli 3.1. Elementi vektorja $\mathcal{P} = [p_j]_{j=1,\ldots,K}$ tako kot pri *osnovnem algoritmu* predstavljajo frekvence črk iz abecede $\mathcal{A}$ v ozadju. Pri algoritmu *Motif Sampler* [47], je vektor ozadja lahko določen iz sekvenc ali pa je podan iz neodvisne množice sekvenc.

Enačbo 3.4 razvijemo s pomočjo Bayesovega pravila po formuli 2.4:

$$\gamma_i(c) = \frac{P(S_i|E_i = c, \mathcal{Q}, \mathcal{P})P(E_i = c|\mathcal{Q}, \mathcal{P})}{P(S_i|\mathcal{Q}, \mathcal{P})} \tag{3.5}$$

Enačbo 3.5 lahko ločimo na tri dele in jih izračunamo posamezno:

1. Vrednost $P(S_i|E_i = c, \mathcal{Q}, \mathcal{P})$ je verjetnost, da je bila sekvenca $S_i$ generirana glede na matriko $\mathcal{Q}$ in vektor $\mathcal{P}$, ki sta določena s $c$ pojavitvami motiva v izbrani sekvenci $S_i$. Izračunamo jo kot vsoto po vseh možnih kombinacijah $c$ motivov v sekvenci $S_i$, in sicer:

$$\begin{aligned} &P(S_i|E_i = c, \mathcal{Q}, \mathcal{P}) = \\ &\quad = \sum_{a_1} \cdots \sum_{a_c} P(S_i|a_1, \ldots, a_c, E_i = c, \mathcal{Q}, \mathcal{P}) P(a_i, \ldots, a_c|E_i = c, \mathcal{Q}, \mathcal{P}), \end{aligned} \tag{3.6}$$

kjer so $a_1, \ldots, a_c$ začetni položaji $c$ pojavitev motiva. Za apriorno porazdelitev $P(a_1, \ldots, a_c | E_i = c, \mathcal{Q}, \mathcal{P})$ predpostavljamo, da je neodvisna od $\mathcal{Q}$ in $\mathcal{P}$. Po tej predpostavki je tako

$$P(a_1, \ldots, a_c | E_i = c) = \frac{1}{\binom{L_x}{c}},$$

kjer je $L_x$ število vseh možnih segmentov $x$ dolžine $W$ v sekvenci dolžine $L$. Zapišimo sekvenco v obliki $S_i = b_1 b_2 \ldots b_{L-1} b_L$, kjer so znaki $b_t \in \mathcal{A}$ za $t = 1, \ldots, L$. Potem prvo verjetnost v 3.6 izračunamo po formuli

$$P(S_i | a_1, \ldots, a_c, E_i = c, \mathcal{Q}, \mathcal{P}) = \\ = \prod_{t=1}^{a_1-1} p_{b_t} \prod_{t=a_1}^{a_1+W-1} q_{t,b_t} \prod_{t=a_1+W}^{a_2-1} p_{b_t} \prod_{t=a_2}^{a_2+W-1} q_{t,b_t} \ \ldots \prod_{t=a_c}^{a_c+W-1} q_{t,b_t} \prod_{t=a_c+W}^{L} p_{b_t}.$$

2. Za apriorno verjetnost $P(E_i = c | \mathcal{Q}, \mathcal{P})$ predpostavimo, da je neodvisna od modela motiva $\mathcal{Q}$ in modela ozadja $\mathcal{P}$. Zato lahko nastavimo

$$P(E_i = c | \mathcal{Q}, \mathcal{P}) = P(E_i = c).$$

3. Verjetnost $P(S_i | \mathcal{Q}, \mathcal{P})$ poračunamo kot vsoto

$$P(S_i | \mathcal{Q}, \mathcal{P}) = \sum_{c=0}^{\infty} P(S_i | E_i = c, \mathcal{Q}, \mathcal{P}) P(E_i = c | \mathcal{Q}, \mathcal{P}).$$

Iz praktičnega vidika je zgornja formula računsko prezahtevna, zato so avtorji predlagali, da se oceni glede na parameter $c_{\max}$, ki označuje maksimalno možno število pojavitev motiva na sekvenco v množici $S$. Potem velja:

$$P(S_i | \mathcal{Q}, \mathcal{P}) = \sum_{c=0}^{c_{\max}} P(S_i | E_i = c, \mathcal{Q}, \mathcal{P}) P(E_i = c | \mathcal{Q}, \mathcal{P}). \tag{3.7}$$

Pričakovano število pojavitev motiva v sekvenci $S_i$ dobimo po formuli

$$\begin{aligned} \mathbb{E}_{(S_i, \mathcal{Q}, \mathcal{P})}[E_i] &= \sum_{c=0}^{c_{\max}} c \cdot P(E_i = c | S_i, \mathcal{Q}, \mathcal{P}) \\ &= \sum_{c=1}^{c_{\max}} c \cdot \gamma_i(c). \end{aligned} \tag{3.8}$$

Vrednosti $\gamma_i(c)$ za $c = 1, \ldots, c_{\max}$ določajo porazdelitev motiva po sekvenci $i$ za različne vrednosti $c$, ki jo označimo z

$$\Gamma_i = \{\gamma_i(c) | c = 1, \ldots, c_{\max}\}.$$

Za vse sekvence v množici $S$ je porazdelitev enaka

$$\Gamma = \{\Gamma_i | i = 1, \ldots, N\}.$$

**Algoritem 6** Psevdokoda iskanja porazdelitve $\Gamma$ v algoritmu *Motif Sampler*.

**Vhod:** Množica $S = \{S_1, \ldots, S_N\}$ sekvenc DNA in pričakovana dolžina motiva $W$.
**Izhod:** Porazdelitev $\Gamma$.

```
1: Ustvari seznam Γ.
2: for i = 1 : N do
3:    for c = 1 : c_max do
4:       Poračunaj γ_i(c).
5:    end for
6:    Vstavi (Γ, γ_i(c)).
7: end for
```

**vrni** $\Gamma$.

Način izračuna porazdelitvene funkcije $\Gamma$ smo za algoritem *GraphGibbs* prilagodili, tako da smo vključili dodatno informacijo o motivu, ki jo dobimo s prvim delom v algoritmu *GraphGibbs*. S pomočjo poravnave $A(y)$ motiva $y$ iz prvega dela lahko določimo parameter $c_{\max}$, čigar izbira je pomembna pri računanju vrednosti $\gamma_i(c)$. Večjo vrednost ima, večja je računska zahtevnost algoritma 6. Pri algoritmu *Motif Sampler* je vrednost $c_{\max}$ sicer poljubna, ampak so avtorji v svojem članku [47] pokazali, da informacijska vrednost statistike $I$ z večanjem vrednosti $c_{\max}$ pada.

Pri algoritmu *GraphGibbs* sta za porazdelitvi pričakovanega števila pojavitev motiva $\widetilde{\boldsymbol{E}} = (E_i | i = 1, \ldots, N)$ možni dve nastavitvi. Lahko nastavimo enakomerno porazdelitev kot v algoritmu 4 in privzamemo, da velja $E_i = E_j$, za vse pare $i, j = 1, \ldots, N$. Pri drugi nastavitvi predvidevamo, da so pojavitve motiva neenakomerno porazdeljene po celi množici. Dodatna predpostavka pri tej nastavitvi je, da je vsaj ena pojavitev motiva v vsaki od danih sekvenc. Razlogi za to predpostavko in možni problemi so izpostavljeni v poglavju 6.

Pri obeh nastavitvah se parameter $c_{\max}$ določi kot najvišji modus v frekvenčni porazdelitvi kopij motiva $y$ glede na njegovo porazdelitev $A(y)$. Poglejmo si primer, ko imamo $N = 7$, število vseh pojavitev motiva je enako $|A(y)| = 6$ in porazdeljeno $\tilde{c}_{A(y)} = \{1, 3, 2, 0, 0, 0, 0\}$. V prvi sekvenci imamo torej eno pojavitev motiva, v drugi tri in tako dalje. Preden pogledamo modus, pa moramo nastaviti vse ničelne vrednosti na 1, saj množice $S$ obravnavamo s predpostavko, da je v vsaki sekvenci vsaj ena pojavitev motiva.

Porazdelitev pojavitev je sedaj enaka $\tilde{c}_{A(y)} = \{1, 3, 2, 1, 1, 1, 1\}$ z modusom $c_{\max} = 1$. Glede na porazdelitev pojavitev $\tilde{c}_{A(y)}$ naredimo še porazdelitev števila pojavitev posamezne vrednosti v $\tilde{c}_{A(y)}$, torej $k = \{num[\tilde{c}_{A(y)}](c) | c = 1, \ldots, C\}$, kjer je $C = \max\{\tilde{c}_{A(y)}\}$. Tako je $k = \{5, 1, 1\}$. Modus smo izbrali, ker najbolje ohranja razmerje med $c_{\max}$ in statistiko $I$. Maksimum $C = 3$ bi lahko namreč postavil degenerirano poravnavo za Gibbsovo vzorčenje, kar se bo odražalo v statistiki $I$. Prav tako se je pri empiričnih poskusih pokazalo, da je maksimalna vrednost v $\tilde{c}_{A(y)}$ prej odstopanje kot pravilo. Povprečje $\tilde{c}_{A(y)}$ pa je občutljivo na ekstremne vrednosti in tako lahko

$c_{\max}$ pride večji, kot bi bilo optimalno.

Vrednost $c_{\max}$ določa poravnavo, ki pa naprej določa model motiva, to je matriko $\mathcal{Q}$. Več "nepravih" vzorčnih nizov izberemo, bolj si bodo blizu vrednosti v matriki $\mathcal{Q}$, kar pa pomeni, da ne moremo z večjo verjetnostjo določiti presečnega vzorca (motiva), saj so vse črke na vseh položajih v motivu enako verjetne. Tako degenerirana matrika $\mathcal{Q}$ nam ne more omogočiti dobre klasifikacije vzorčnih množic.

Poleg parametra $c_{\max}$ si z algoritmom *GraphGibbs* lahko pomagamo pri določanju verjetnosti $P(E_i = c)$, da ni več enakomerno določena. Poglejmo na iskanje števila pojavitev motiva z vidika Bernoullijevih poskusov. Število vseh poskusov je sedaj enako $n = N \cdot C$, ki je tudi maksimalno število pričakovanih pojavitev poskusa glede na rezultat prvega dela našega algoritma. Uspešni zadetki $\boldsymbol{z}$ so porazdeljeni kot $A(y)$. Želimo določiti kakšni so pričakovani deleži pojavitev motiva na sekvenco $\theta_c = \frac{k_c}{n}$, za vsak $k_c \in k$. Le-te zberemo v slučajni vektor $\boldsymbol{\theta} = (\theta_c)_{c=1,\ldots,C}$.

Naj $\theta$ predstavlja poljuben delež $\theta_i$ in $z = \psi(y)$. Bernoullijeve poskuse lahko naravno modeliramo z binomsko porazdelitvijo [14], in sicer:

$$p(z|\theta) \propto \theta^z (1-\theta)^{n-z}.$$

Z uporabo Bayesovega pravila po formuli 2.4 določimo porazdelitev $p(\theta|z)$ sorazmerno kot

$$p(\theta|z) \propto p(z|\theta)p(\theta). \tag{3.9}$$

Za izračun porazdelitve 3.9 je potrebno določiti še $p(\theta)$. Matematično je priročno vzeti apriorno porazdelitev, ki je naravno konjugirana binomski porazdelitvi. Predvidimo torej, da je $p(\theta) \sim \text{Beta}(\alpha, \beta)$. Potem velja $p(\theta|z) \sim \text{Beta}(\alpha + z, \beta + n - z)$.

Določiti moramo še hiperparametra $\alpha > 0$ in $\beta > 0$. Za $\theta \sim \text{Beta}(\alpha, \beta)$ velja:

$$\mathbb{E}[\theta] = \frac{\alpha}{\alpha + \beta} \quad \text{in}$$
$$\text{var}[\theta] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}.$$

Označimo $\mathbb{E}[\theta] = a$ in $\text{var}[\theta] = b$, da dobimo

$$\alpha = \frac{(1-a)a - ab}{b},$$
$$\beta = \frac{((1-a)a - b)(1-a)}{b}. \tag{3.10}$$

Vrednosti $a$ in $b$ določimo glede na odgovarjajoči vrednosti uteženega povprečja in standardne deviacije, kjer so uteži enake za vseh $n$ opazovanj, in sicer $u_i = \frac{1}{C}$. Naj bo $\boldsymbol{c} = \tilde{c}_{A(y)}$ in utežen vektor $\boldsymbol{c}_u = \{u_i \cdot c_i\}_{i=1}^N$. Potem definiramo:

$$a = \overline{\boldsymbol{c}}_u = \frac{\overline{\boldsymbol{c}}}{C} = \frac{\frac{1}{N}\sum_{i=1}^N c_i}{C},$$
$$b = \sigma^2(\boldsymbol{c}_u) = \frac{\sigma^2(\boldsymbol{c})}{C^2} = \frac{\frac{1}{N-1}\sum_{i=1}^N (c_i - \overline{\boldsymbol{c}})^2}{C^2}. \tag{3.11}$$

Vrednosti $a$ in $b$ iz 3.11 vstavimo v 3.10, da določimo beta porazdelitev. Hiperparametra $\alpha$ in $\beta$ morata biti pozitivna. Vrednosti $a$ in $b$ sta nenegativni števili. Problemi pri izračunu 3.10 lahko nastopijo, takrat ko je:

1. $c_i = \overline{\boldsymbol{c}}$ za vsak $i$ ali
2. $\overline{\boldsymbol{c}} = C$.

V prvem primeru so vsi elementi enaki povprečni vrednosti, kar pomeni, da ni disperzije oziroma $b = 0$. Potlej sta vrednosti $\alpha$ in $\beta$ nedefinirani. V drugem primeru pa je povprečna vrednost enaka največji vrednosti $C$ v $\boldsymbol{c}$ in tako je $a = 1$. Zato sta vrednosti $\alpha$ in $\beta$ enaki nič. Oba primera se zgodita, ko imamo enakomerno porazdeljene uspešne poskuse v $A(y)$. V tem primeru namesto porazdelitve Beta vzamemo enakomerno porazdelitev za $P(E_i = c)$, kar je tudi bolj intuitivno in skladno z dodatno informacijo, ki smo jo dobili iz prvega dela.

Na gornji način uporabimo dodatno informacijo o $A(y)$ ne samo za določitev začetka iteracije pri Gibbsovem vzorčevalniku, ampak tudi za izračun informativne apriorne porazdelitve za $P(E_i = c)$. Za namen določevanja števila pojavitev v posamezni sekvenci pri predpostavki, da so le-ta neenakomerno porazdeljena po sekvencah, nam informativna apriorna porazdelitev dobljena po 3.9 bolje uteži izračun 3.5.

Skupaj s $\boldsymbol{c}$ definiramo vektor pričakovanega števila pojavitev $\boldsymbol{\xi} = (\xi_i)_{i=1,\dots,N}$, kjer elemente določimo kot

$$\xi_i = \max\left\{\mathbb{E}_{(S_i,\mathcal{Q},\mathcal{P})}[E_i], c_i\right\}.$$

Parameter $\boldsymbol{\xi}$ upoštevamo naprej pri Gibbsovem vzorčenju. Na polovici iteracij ponovno določimo porazdelitev $\Gamma$ in vektor $\boldsymbol{\xi}$ z vso informacijo, ki jo imamo na razpolago do tistega trenutka. S tem preverimo, ali so potrebne spremembe vrednosti $\xi_i$ za nadaljnjo obdelavo. Vključno s to modifikacijo je algoritem *GraphGibbs* opisan v psevdokodi 7.

**Algoritem 7** Psevdokoda algoritma *GraphGibbs*.

**Vhod:** Množica $S = \{S_1, \ldots, S_N\}$ sekvenc DNA.
**Izhod:** Motiv z odgovarjajočo poravnavo $A(\cdot)$.

Združi sekvence $S_i$ v eno sekvenco $R$.
Numerično kodiraj sekvence $S_i$ s trojčki.
**repeat**
  Preberi sekvenco $R$ in ustvari matriko povezav.
  Ustvari matriko sosednosti.
  Poišči vse sprehode od desetih izbranih vozlišč.
  Vzemi vozlišča z največjo vhodno stopnjo.
  Sestavi motiv od začetnega šestorčka.
  Odstrani dobljeni motiv iz vhodnih sekvenc.
**until** ni več motivov ali $M$

**for all** najden motiv $y$ iz prvega dela **do**
  **if** predpostavimo enakomerno porazdelitev **then**
    $E_i = c_{max}\ \forall\ i = 1, \ldots, N$
  **else**
    Poračunaj hiperparametra $\alpha$ in $\beta$.
    Določi porazdelitev $\Gamma$.
    Izračunaj vektor števila pojavitev $\boldsymbol{\xi}$.
    **for all** $i = 1 : N$ **do**
      $E_i = \xi_i$
    **end for**
  **end if**

  Vzemi najdeno poravnavo $A(y) = \{\boldsymbol{a_1}, \ldots, \boldsymbol{a_N}\}$ in jo slučajno dopolni.
  **repeat**
    Izberi sekvenco $S_z$ in jo izloči iz $S$, skupaj z njenim položajem $\boldsymbol{a_z}$ v $A(y)$.
    Izračunaj $\mathcal{Q}$ in $\mathcal{P}$ iz $S \backslash \{S_z\}$.
    Uteži vse segmente $x^p$, $p = 1, \ldots, L'_z$, v sekvenci $S_z$ z utežmi $A_{x^p}$.
    Vzemi segmente $\{x^{p_j}\}_{j=1}^{\xi_z}$ slučajno in določi nove vrednosti $a_{zj} = p_j$.
  **until** ni dosežena konvergenca ali $M$
**end for**

**vrni** poravnavo $A(y)$ za vsak najden motiv $y$

# 4. Obdelava

Uspešnost algoritma *GraphGibbs* smo preverili tako na realnih množicah kot na generiranih podatkovnih množicah, ki simulirajo DNA sekvence. Pred analizo uspešnosti algoritma smo naredili še analizo konvergence in Plackett-Burmanov načrt (ang. Plackett-Burman factorial design) za testiranje vplivnih dejavnikov pri izvajanju algoritma.

## 4.1 Podatki

Algoritem *GraphGibbs* smo testirali na več različnih podatkovnih množicah. Uporabili smo tako realne podatke kot tudi simulirane podatke z znanimi motivi. Dodatno smo generirali tako imenovane testne množice, ki smo jih uporabili pri Plackett-Burmanovem eksperimentalnem načrtu. S tem smo se izognili prevelikemu prilagajanju modela pri optimizaciji algoritma *GraphGibbs*.

### Generirani podatki

Generirali smo lastne množice podatkov, saj te predstavljajo dobro kontrolirano eksperimentalno okolje za ocenjevanje uspešnosti algoritma oziroma metode. Ker so položaji motivov znani, se vrednosti evalvacijskih statistik, kot sta občutljivost in pozitivna napovedna vrednost, lahko natančno izračunajo. Pri realnih množicah so izračuni lahko manj natančni, ker so znani motivi določeni eksperimentalno.

Tabela 4.1: Osnovni parametri za generiranje dodatnih množic z znanimi motivi.

| **parameter** | **oznaka** | **vrednosti** |
|---:|:---:|:---|
| število motivov | $M$ | $1, 2$ |
| dolžina motiva | $W$ | $6, 9, 13, 18$ |
| stopnja variabilnosti | $dv$ | $0, 1, 2, 3$ |
| porazdeljenost motivov | $P$ | $u, nu$ |
| število pričakovanih pojavitev motiva | $E$ | $1, 2$ |
| število sekvenc | $N$ | $5, 10, 15$ |
| dolžina sekvence (ozadje) | $L$ | $100, 500, 1000, 2500$ |

Na podlagi lastnosti množic realnih podatkov smo simulirali dodatne množice podatkov. Določili smo razpon za število sekvenc v množici, za dolžino posamezne sekvence in število vzorčnih nizov (oziroma število pričakovanih pojavitev motiva). Vrednosti slednjega števila interpretiramo kot eno pojavitev motiva na sekvenco ($E = 1$) ali dve pojavitvi motiva na sekvenco ($E = 2$) v primeru enakomerne porazdelitve in kot $E = N$ pojavitev motiva v množici ali $E = 2N$ pojavitev motiva v množici v primeru neenakomerne porazdelitve.

Pomembna razlika je v poenotenju dolžine motiva in v manjši stopnji variabilnosti. Tako smo izbrali zaradi večje kontrole pri testiranju našega algoritma, kajti neenotna dolžina motiva poveča kompleksnost modela in relativna ohranjenost je ena izmed predpostavk pri reševanju osnovnega problema, kot smo ga opredelili v poglavju 1.2. Dodatni parameter je še porazdelitev vzorčnih nizov po sekvencah, in sicer so porazdeljeni enakomerno po sekvencah ali pa razpršeni po celi množici, torej je porazdelitev neenakomerna. Vsi parametri pri generiranju množic, njihove oznake in vrednosti so podani v tabeli 4.1.

V tabeli 4.1 smo za porazdeljenost motivov po sekvencah uporabili kratici $u$ in $nu$ angleških oznak za enakomerno (ang. uniform) in neenakomerno porazdelitev (ang. non uniform), da se izognemo prekrivanju oznak. Za $M = 1$ dobimo 624 različnih množic. Dodatno smo naredili še 9 množic z dvema motivoma, da prikažemo še delovanje algoritma z različnimi tipi motivov. Te množice so predstavljene v tabeli 4.2, kjer so podane tako osnovne karakteristike ($N$, $L$) množic kot tudi osnovne lastnosti dveh motivov. Skupaj dobimo 633 generiranih množic za preverjanje uspešnosti algoritma *GraphGibbs*.

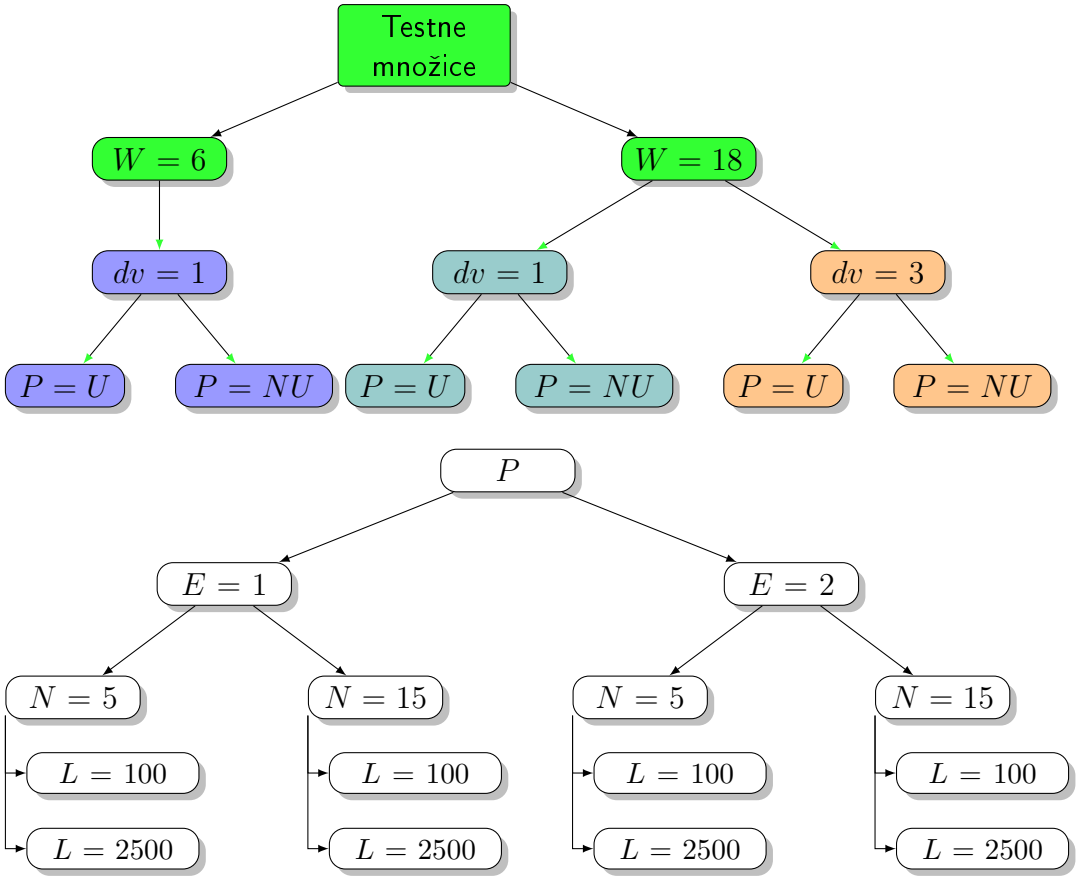
Tabela 4.2: Množice z dvema motivoma in osnovne lastnosti motivov.

| | | | **motiv** 1 | | | | **motiv** 2 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ♯ množice | $N$ | $L$ | $W$ | $dv$ | $E$ | $P$ | $W$ | $dv$ | $E$ | $P$ |
| 1 | 10 | 500 | 6 | 0 | 2 | $u$ | 9 | 0 | 2 | $u$ |
| 2 | 5 | 1000 | 13 | 1 | 1 | $u$ | 6 | 1 | 2 | $nu$ |
| 3 | 15 | 2500 | 9 | 2 | 1 | $u$ | 18 | 0 | 1 | $nu$ |
| 4 | 5 | 100 | 6 | 0 | 2 | $u$ | 13 | 2 | 1 | $u$ |
| 5 | 15 | 1000 | 18 | 2 | 1 | $nu$ | 9 | 2 | 2 | $nu$ |
| 6 | 10 | 500 | 13 | 2 | 1 | $u$ | 18 | 1 | 2 | $u$ |
| 7 | 10 | 100 | 18 | 3 | 1 | $u$ | 9 | 1 | 2 | $nu$ |
| 8 | 5 | 500 | 13 | 3 | 1 | $nu$ | 6 | 0 | 2 | $nu$ |
| 9 | 10 | 2500 | 13 | 3 | 2 | $nu$ | 18 | 3 | 1 | $u$ |

### Testne množice

Posebej smo ustvarili še 48 testnih množic. Te smo uporabili pri predpripravi algoritma *GraphGibbs*, in sicer za izvedbo Plackett-Burmanovega načrta in analizo konvergence, ker istih množic ne moremo uporabiti tudi za analizo uspešnosti algoritma. Izogniti se moramo namreč prevelikemu prilagajanju modela na podatke (ang. overfitting).

Zaradi večje kontrole pri obdelavah smo se pri testnih množicah omejili samo na generirane podatke. Vzeli smo ekstremne vrednosti parametrov dolžine motiva $W$, števila sekvenc $N$, dolžino sekvence $L$ in stopnje variabilnosti $dv$. Pri slednjem parametru smo izločili možnost $dv = 0$, saj algoritem *GraphGibbs* najde popolnoma ohranjene motive že v prvem delu in tako ne bi mogli narediti analize na vseh prostih dejavnikih v Plackett-Burmanovem načrtu. Shema vseh testnih množic z omejitvami na parametrih iz tabele 4.1 je prikazana na sliki 4.1.




Slika 4.1: Shema strukture testnih množic glede na parametre iz tabele 4.1. Drugo drevo je nadaljevanje prvega drevesa iz vsakega lista.

### Realni podatki

Realne množice podatkov smo dobili iz prosto dostopne primerjalne zbirke množic podatkov, ki so jo sestavili Martin Tompa s sodelavci [48]. Na teh podatkih so analizirali 13 različnih algoritmov za iskanje motivov. Sestavili so podatkovne množice z realnimi sekvencami, ki so jih dobili iz TRANSFAC podatkovne baze. Uporabili so sekvenčne dele DNA štirih vrst, in sicer: glive, muhe, miši in človeka. Skupaj so dobili 52 množic.

Tabela 4.3: Osnovne lastnosti množic realnih DNA sekvenc štirih živalskih vrst.

| | **človek** | | | **miš** | | | **gliva** | | | **muha** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S$ | $N$ | $m$ | $L$ | $N$ | $m$ | $L$ | $N$ | $m$ | $L$ | $N$ | $m$ | $L$ |
| 1 | 18 | 16 | 2000 | 3 | 6 | 500 | 9 | 7 | 1000 | 4 | 7 | 1500 |
| 2 | 9 | 11 | 1000 | 9 | 12 | 1000 | 4 | 5 | 500 | 1 | 5 | 2000 |
| 3 | 10 | 15 | 1500 | 5 | 9 | 500 | 8 | 18 | 500 | 3 | 9 | 2000 |
| 4 | 13 | 11 | 2000 | 7 | 14 | 1000 | 7 | 7 | 1000 | 4 | 9 | 2000 |
| 5 | 3 | 11 | 1000 | 4 | 6 | 500 | 3 | 4 | 500 | 3 | 14 | 2500 |
| 6 | 9 | 9 | 500 | 3 | 5 | 500 | 7 | 7 | 500 | 1 | 7 | 3000 |
| 7 | 5 | 6 | 1000 | 4 | 4 | 1500 | 11 | 14 | 1000 | | | |
| 8 | 15 | 13 | 500 | 3 | 3 | 1500 | 16 | 13 | 1000 | | | |
| 9 | 10 | 10 | 1500 | 2 | 2 | 1500 | | | | | | |
| 10 | 6 | 11 | 500 | 12 | 14 | 1000 | | | | | | |
| 11 | 8 | 19 | 1000 | 12 | 15 | 500 | | | | | | |
| 12 | 2 | 5 | 500 | 3 | 7 | 500 | | | | | | |
| 13 | 6 | 9 | 1000 | | | | | | | | | |
| 14 | 2 | 4 | 1000 | | | | | | | | | |
| 15 | 4 | 4 | 2000 | | | | | | | | | |
| 16 | 7 | 7 | 3000 | | | | | | | | | |
| 17 | 11 | 10 | 500 | | | | | | | | | |
| 18 | 5 | 7 | 3000 | | | | | | | | | |
| 19 | 5 | 4 | 500 | | | | | | | | | |
| 20 | 35 | 76 | 2000 | | | | | | | | | |
| 21 | 5 | 7 | 1000 | | | | | | | | | |
| 22 | 6 | 5 | 500 | | | | | | | | | |
| 23 | 4 | 5 | 500 | | | | | | | | | |
| 24 | 8 | 8 | 500 | | | | | | | | | |
| 25 | 2 | 5 | 500 | | | | | | | | | |
| 26 | 9 | 10 | 1000 | | | | | | | | | |

$S$ - množica podatkov, $N$ - število sekvenc, $m$ - število vzorčnih nizov, $L$ - dolžina sekvence

Osnovne lastnosti množic z realnimi podatki so podane v tabeli 4.3. Prvi stolpec tabele predstavlja zaporedno številko množice. Vrstni red je povzet iz baze. Za vsako živalsko vrsto so podane tri lastnosti, in sicer število sekvenc v množici ($N$), število vseh znanih vzorčnih nizov ($m$) in celotna dolžina vsake sekvence v množici ($L$). Vzorčni nizi imajo visoko stopnjo variabilnosti in so različnih dolžin. To predstavlja dodatne omejitve pri obdelavi in interpretaciji rezultatov, kar je podrobneje opisano v poglavju 5.

## 4.2 Statistične obdelave

Statistične obdelave za preverjanje uspešnosti delovanja algoritma *GraphGibbs* smo izvedli v dveh korakih. Najprej smo vplivne dejavnike analizirali s pomočjo faktorialnega načrta in naredili analizo konvergence Gibbsovega vzorčevalnika. Z obema analizama smo izboljšali delovanje algoritma, tako da smo določili vrednosti parametrov za obdelave podatkovnih množic. V drugem koraku smo uspešnost rezultatov preverili z evaluacijskimi statistikami, ki so opisane v podpoglavju 4.2.2.

### 4.2.1 Optimizacija algoritma *GraphGibbs*

Za izboljšanje delovanja algoritma smo naredili Plackett-Burmanov načrt vplivnih dejavnikov, kjer smo preverili, kateri prosti parametri algoritma imajo pomemben vpliv na njegovo delovanje. Ta načrt smo izvedli na ločenih testnih množicah. Dodatno smo naredili še analizo konvergence.

V modelu uporabljamo 7 prostih parametrov:

1. število motivov (ki jih pričakujemo v množici),
2. dolžine motivov,
3. število pričakovanih pojavitev motiva,
4. porazdelitev števila pojavitev motiva,
5. število iteracij Gibbsovega vzorčevalnika,
6. funkcija za izračun psevdo števil (funkcija $B$ v 3.1), in
7. orientacija sekvenc.

Parametre $1, 2, 3$ in 7 smo prevzeli iz osnovnega algoritma, saj je pomembno, da ima uporabnik možnost določiti, koliko motivov išče v množici, kakšnih dolžin so ti motivi in kolikokrat se pojavijo na sekvenco. S parametrom 7 (orientacija sekvenc) določimo, katero verigo dvojne vijačnice uporabimo.

Drugi parametri so posledica razvijanja metode. Ko smo v naš algoritem integrirali še iskanje porazdelitve pričakovanih pojavitev motiva, smo kot prosti parameter določili še parameter 4, ki ima dve stanji: enakomerno porazdeljene (enako število pojavitev na sekvenco) in neenakomerno porazdeljene pričakovane pojavitve motiva po celi množici. Ena izmed predpostavk naše metode je, da se iskani motiv pojavi vsaj enkrat na sekvenco, vendar obstaja, za razliko od osnovnega algoritma, kjer je število pojavitev motiva enako za vse sekvence, pri algoritmu *GraphGibbs* možnost, da so pojavitve iskanega motiva neenakomerno porazdeljene po celi množici.

Druga dva parametra sta pomembna pri določanju končne poravnave. Ustvarjalci osnovnega algoritma [27], so za določanje psevdo števil predlagali funkcijo $B = \sqrt{N}$. Uporabo kvadratnega korena so utemeljili na podlagi empiričnih poskusov, zato smo

pri razvoju algoritma *GraphGibbs* želeli preizkusiti še druge možnosti za funkcijo $B$. Funkcija $B$ ima namreč vpliv na izračun elementov $q_{ij}$ pozicijsko utežene matrike $Q$ po formuli 3.1. Njihove vrednosti naprej določajo velikost ocene $F$, izračune po formuli 3.2. Večji kot so ti elementi, višja je ocena $F$ in posledično je tudi višja vrednost statistike $I$, formula 3.3.

Število iteracij Gibbsovega vzorčevalnika nam pomaga kontrolirati Markovsko verigo, ki jo gradimo pri Gibbsovem vzorčenju. Potrebujemo dovolj veliko število iteracij, da dosežemo stacionarno verigo, saj to nakazuje na ekstrem v prostoru rešitev oziroma v prostoru poravnav. Glede na rešitve prvega dela algoritma *GraphGibbs*, je začetna točka namesto slučajne izbire, ki je lahko oddaljena od ekstremov, deloma določena. Že delna določitev bi morala pospešiti prehod Markovske verige v stacionarno verigo.

Velikost vpliva teh sedmih dejavnikov na uspešnost algoritma *GraphGibbs* smo preverili s Plackett-Burmanovim eksperimentalnim načrtom (okrajšano PB načrt). S PB načrtom smo omejili število prostih dejavnikov na tiste, ki imajo značilen vpliv na izračuna statistike $I$, s katero ocenjujemo poravnavo vzorčnih nizov, ki predstavlja rešitev našega algoritma. Dodatno smo še testirali vpliv posameznih dejavnikov na odstotek ujemanja presečnega vzorca z znanim motivom. Visok odstotek ujemanja lahko dobimo tudi na "nepravih" vzorčnih množicah, torej na množicah, katerih vzorčni nizi niso enaki znanim vzorčnim nizom. Vendar je tudi neprava vzorčna množica z visokim odstotkom ujemanja presečnega vzorca z iskanim motivom lahko pomembna pri analizi realnih podatkov. Pri realnih sekvencah so vzorčni nizi nekega motiva določeni eksperimentalno in tako obstaja možnost, da del vzorčnih nizov ni bil zaznan in tako niso del znane (prave) vzorčne množice. Ti vzorčni nizi teoretično lahko gradijo drugo (nepravo) vzorčno množico, ki pa jo algoritem najde.

### Načrt Plackett-Burman

Dejavnike, ki močno vplivajo na izračun statistike $I$ in odstotek ujemanja, smo določili s Plackett-Burmanovim načrtom. Plackett-Burmanovi načrti spadajo med *delne faktorske eksperimentalne načrte* (ang. fractional factorial design, okrajšano FFD). R.L. Plackett in J.P. Burman sta razvila načrte, kjer je število kombinacij dejavnikov večkratnik števila štiri namesto dva. Razvoj načrtov je bil namenjen testiranju robustnosti procesa ali metode s čim manjšim številom poskusov [11].

Faktorski načrti (ang. factorial designs) in FFD načrti določajo velikost vpliva posameznih dejavnikov oziroma faktorjev na rezultat eksperimenta, pri čemer je obdelava narejena na vseh dejavnikih hkrati. Pri teh metodah se zaznajo tudi interakcije med dejavniki.

Vsak dejavnik v načrtu ima lahko več stanj (nivojev). Izbiramo lahko med nominalnimi, ordinalnimi, intervalskimi in/ali razmernostnimi stanji. Kvalitativni dejavniki imajo nominalna stanja, kvantitativni dejavniki pa ostala. Glede na število dejavnikov $K$ in njihovih stanj so se razvili različni popolni faktorski načrti, na primer tako imenovani $2^K$ in $3^K$ načrti.

V primeru $2^K$ načrtov imamo $K$ dejavnikov, ki imajo po dve možni stanji, in sicer nizko in visoko. Kaj predstavlja posamezno stanje, uporabnik določi glede na sam dejavnik in na naravo eksperimentov. Velikokrat raziskovalci po subjektivni presoji določijo vrednosti, pri katerih pričakujejo spremembo rezultatov. Stanja lahko določijo tudi na osnovi natančnosti ali negotovosti. Pri kvantitativnih dejavnikih za vrednosti teh stanj pogosto vzamemo ekstremni vrednosti, to je najnižjo in najvišjo vrednost, ki jo lahko doseže dejavnik, vendar je interval možnih vrednosti stanja potrebno preveriti za vsak dejavnik posebej in v kontekstu eksperimentalnega načrta [19].

Na podlagi števila stanj sestavimo *matriko načrta* (ang. design matrix), v kateri so predstavljena vsa stanja vseh dejavnikov [3]. Za zgled vzemimo $2^K$ načrt za dva dejavnika $X_1$ in $X_2$. Matrika načrta je potem oblike:

$$D = \begin{bmatrix} X_{11} & X_{21} \\ X_{11} & X_{22} \\ X_{12} & X_{21} \\ X_{12} & X_{22} \end{bmatrix} \tag{4.1}$$

Posamezno stanje označimo z dodatnim indeksom, torej $X_{11}$ je prvo stanje prvega dejavnika. Vsaka vrstica v matriki $D$ predstavlja *eksperimentalno kombinacijo* (ang. treatment combination [3]) ali drugače, kombinacijo stanj dejavnikov, ki jo obravnavamo pri enem poskusu. Število vrstic v matriki $D$ je tako enako številu potrebnih poskusov $n$, ki jih potrebujemo za obdelavo vpliva dejavnikov. V navedem primeru je to enako produktu števila stanj posameznih dejavnikov, torej $n = 2 \cdot 2 = 4$.

Pri eksperimentalni kombinaciji dobimo *rezultat* ali *odgovor* (ang. response) $Y$. Ko imamo $n$ kombinacij, dobimo $n$ rezultatov $Y_i$, $i = 1, \ldots, n$. Rezultat obravnavamo kot slučajno spremenljivko in jo zapišemo z vsoto pričakovane vrednosti in napake, torej kot

$$Y_i = \mathbb{E}[Y_i] + e_i.$$

Odvisnost rezultata $Y_i$ od dejavnikov $X_j$, $j = 1, 2$ lahko zapišemo v obliki enačbe prvega reda:

$$Y_i = B_0 + B_1 X_{1i} + B_2 X_{2i} + e_i, \quad i = 1, \ldots, n, \tag{4.2}$$

kjer so $B_0, B_1$ in $B_2$ neznane konstante (parametri modela) in $e_i$ predstavlja napako meritve. V enačbi prvega reda tako dobimo konstantni člen (ang. intercept) $B_0$, vpliv prvega reda dejavnika $X_1$ in vpliv prvega reda dejavnika $X_2$. V modelu prvega reda nismo upoštevali interakcije $B_{12}$ med dvema dejavnikoma. Ta člen je vključen v modelu drugega reda, to je:

$$Y_i = B_0 + B_1 X_{1i} + B_2 X_{2i} + B_{12} X_{1i} X_{2i} + e_i, \quad i = 1, \ldots, n.$$

Vpliv dejavnika $B_{12}$ na rezultat $Y_i$ je posledica spremembe stanja enega dejavnika ($X_2$) zaradi spremembe stanja drugega dejavnika ($X_1$). V PB načrtu se obravnavajo samo glavni vplivi (ang. main effects) brez interakcij med njimi. Zato bo za nadaljnjo razpravo pomembna predvsem modelna enačba 4.2.

Alternativno lahko model 4.2 predstavimo v matrični obliki. Najprej razširimo matriko načrta $D$ v matriko $X = [\mathbf{1}, D]$, tako da dodamo stolpec samih enic matriki $D$ za izračun konstantnih členov v modelu. Matrična oblika za model prvega reda:

$$\mathbb{E}[Y_i] = \boldsymbol{X_i} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \end{bmatrix} = \boldsymbol{X_i B}$$

$$\forall i: \quad \begin{bmatrix} \mathbb{E}[Y_1] \\ \mathbb{E}[Y_2] \\ \vdots \\ \mathbb{E}[Y_n] \end{bmatrix} = \begin{bmatrix} \boldsymbol{X_1 B} \\ \boldsymbol{X_2 B} \\ \vdots \\ \boldsymbol{X_n B} \end{bmatrix} = X\boldsymbol{B}$$

$$\text{oziroma } \mathbb{E}[\boldsymbol{Y}] = X\boldsymbol{B}.$$

Matrika $X$ se imenuje *matrika modela* (ang. model matrix) dimenzij $n \times n$.

Parametri modela $\boldsymbol{B} = \{B_0, B_1, B_2\}$ so neznani, zato vzamemo približek $\boldsymbol{B} \approx \boldsymbol{b}$, ki ga izračunamo po metodi najmanjših kvadratov [11]. V matrični obliki tako dobimo:

$$\boldsymbol{b} = (X'X)^{-1} X'Y. \tag{4.3}$$

Vektor $\boldsymbol{b}$ je linearna kombinacija dobljenih rezultatov. Če predpostavimo, da so napake v modelu 4.2 neodvisne z varianco $Var(\boldsymbol{e}) = I\sigma^2$. Potem je varianca ocene $\boldsymbol{b}$ enaka

$$Var(\boldsymbol{b}) = (X'X)^{-1} \sigma^2.$$

Posebno so zanimivi tako imenovani *ortogonalni načrti* (ang. orthogonal design), kadar ima matrika $X'X$ neničelne elemente samo na diagonali. To lastnost imajo tudi $2^K$ načrti, kjer velja

$$(X'X)^{-1} = \frac{1}{n} I.$$

Popolni faktorski načrti imajo to slabost, da se število eksperimentalnih kombinacij povečuje z povečevanjem števila dejavnikov, in sicer

$$\begin{array}{rcccccc} K & = & 2, & 3, & 4, & 5, & 6, & 7 \\ N & = & 4, & 8, & 16, & 32, & 64, & 128 \end{array}.$$

Prevelikemu številu eksperimentov se lahko izognemo, če nekaj dejavnikov fiksiramo in uporabimo manjši eksperimentalni načrt za pregled ostalih dejavnikov. A smisel uporabe eksperimentalnega načrta je ravno pregled vpliv vseh dejavnikov hkrati. V ta namen so se razvili FFD načrti, kjer še vedno naredimo pregled vseh dejavnikov istočasno, vendar predpostavimo, da so višje interakcije med dejavniki zanemarljive. Z večanjem števila dejavnikov se veča tudi število interakcij med dvema in več dejavniki. V tabeli 4.4 je prikazano število interakcij dveh in višjih redov za števila dejavnikov $K \leq 7$.

Tabela 4.4: Prikaz števila interakcij med dvema in več dejavniki glede na njihovo število $K$.

| **Dejavniki** | $\boldsymbol{K=2}$ | $\boldsymbol{K=3}$ | $\boldsymbol{K=4}$ | $\boldsymbol{K=5}$ | $\boldsymbol{K=6}$ | $\boldsymbol{K=7}$ |
|---|---|---|---|---|---|---|
| konst. člen | 1 | 1 | 1 | 1 | 1 | 1 |
| glavni | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 FI | 1 | 3 | 6 | 10 | 15 | 21 |
| 3 FI | | 1 | 4 | 10 | 20 | 35 |
| 4 FI | | | 1 | 5 | 15 | 35 |
| 5 FI | | | | 1 | 6 | 21 |
| 6 FI | | | | | 1 | 7 |
| 7 FI | | | | | | 1 |

FI = interakcija med dejavniki (ang. Factor Interaction)

Pri FFD načrtih se interakcije višjih redov *združijo* (ang. confound/alie) z glavnimi vplivi in/ali z interakcijami nižjega reda. Glede na število (red) zanemarljivih vplivov imajo FFD načtri sodo ali liho *resolucijo* (ang. resolution), ki jo označimo z rimskimi številkami. Red resolucije glede na red opaznih vplivov (OV) in red zanemarljivih vplivov (ZV) je prikazan v tabeli 4.5.

Tabela 4.5: Resolucija FFD načrta glede število opaznih vplivov (OV) in zanemarljivih vplivov (ZV).

| **Resolucija** | **Število OV** | **Število ZV** |
|---|---|---|
| $2t+1$ | $t$ ali manj | $t+1$ ali več |
| $2t$ | $t-1$ ali manj | $t+1$ ali več |

Resolucija torej določi število zanemarljivih vpliv oziroma koliko interakcij višjih redov zanemarimo. Za izvedbo načrta z nižjo resolucijo je potrebnih manj eksperimentalnih kombinacij ($N$) glede na število dejavnikov $K$. Na primer, najmanjše število eksperimentalnih kombinacij za FFD načrt resolucije III s $K$ dejavniki je $N = K + 1$. Kadar je število eksperimentalnih kombinacij večje od števila dejavnikov samo za eno, govorimo o *nasičenih načrtih* (ang. saturated designs).

Uvedimo oznako $2^K//N$, ki označuje $N$-ti delež polnega $2^K$ načrta. V primeru, da je $N = K + 1$ večkratnik števila 4, potem uporabimo načrt iz razreda ortogonalnih nasičenih načrtov resolucije III, poznani tudi pod imenom Plackett-Burman načrti. Prve vrstice v matriki načrta $D$ *kodiranega* PB načrta so podane v tabeli 4.6, kjer + predstavlja visoko stanje in − predstavlja nizko stanje.

S kodiranjem stanj z znakoma + in − načrt postane simetričen, kar se zrcali tudi pri izračunu cenilke $\boldsymbol{b}$. Simetrija kodiranega načrta ima predvsem pomembne matematične prednosti, kot je izračun vplivov dejavnikov, torej parametrov modela, v PB načrtu. Podrobnosti, kot je prehod iz kodiranih v nekodirane ocene parametrov, si lahko bralec ogleda v [11]. V nadaljevanju bomo obravnavali kodiran PB načrt,

Tabela 4.6: Prva vrstica matrike načrta $D$ v kodiranem Plackett-Burman načrtu pri različnem številu dejavnikov $K$.

| $K$ | $N$ | Prva vrstica matrike načrta $D$ |
|---|---|---|
| 7 | 8 | + + + − + − − |
| 11 | 12 | + + − + + + − − − + − |
| 15 | 16 | + + + + − + − + + − − + − − − |
| 19 | 20 | + + − − + + + + − + − + − − − − − + + − |
| 23 | 24 | + + + + + − + − + + − − + + − − + − + − − − − |

ki smo ga tudi uporabili za analizo prostih parametrov v algoritmu *GraphGibbs*.

V tem delu smo uporabili *minimalen* PB načrt, kar pomeni, da smo vzeli načrt brez tako imenovanih *dodatnih spremenljivk* (ang. dummy variables). Te spremenljivke so imaginarni dejavniki, katerih spremembe stanj nimajo fizičnega pomena. Dodatne spremenljivke so potrebne, kadar nimamo dovolj dejavnikov definiranih, da lahko izpeljemo nasičen PB načrt. Pri algoritmu *GraphGibbs* smo definirali sedem opaznih dejavnikov, ki vplivajo na izhod algoritma. Da dobimo nasičen PB načrt, potrebujemo $N = 8$ ekperimentalnih kombinacij, kar pa je tudi najmanjše možno število pri tem številu dejavnikov. Od tod izraz minimalen PB načrt.

Ker ima PB načrt resolucijo III, so vse interakcije dveh in višjih redov zanemarjene, kar je razvidno iz tabele 4.5. Vplivi teh interakcij se tako združijo z glavnimi vplivi dejavnikov. Sledimo modelu prvega reda 4.2, kjer so združeni naslednji parametri modela pri ocenjevanju cenilke $\boldsymbol{b} = \{b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7\}$, po [3]:

$$\begin{aligned}
\mathbb{E}[b_0] &= B_0, & \mathbb{E}[b_4] &= B_4 - B_{13} - B_{25} - B_{67},\\
\mathbb{E}[b_1] &= B_1 - B_{26} - B_{34} - B_{57}, & \mathbb{E}[b_5] &= B_5 - B_{17} - B_{24} - B_{36},\\
\mathbb{E}[b_2] &= B_2 - B_{16} - B_{37} - B_{45}, & \mathbb{E}[b_6] &= B_6 - B_{12} - B_{35} - B_{47},\\
\mathbb{E}[b_3] &= B_3 - B_{14} - B_{27} - B_{56}, & \mathbb{E}[b_7] &= B_7 - B_{15} - B_{23} - B_{46},
\end{aligned}$$

Prva vrstica matrike načrta $D$ za PB načrt uporabljen v tem delu je določena po tabeli 4.6. Ostale vrstice $i = 2, \ldots, 7$ dobimo z cikličnim premikanjem prve vrstice v desno za $i - 1$ korakov. Zadnja vrstica vsebuje samo $-$ znake. Tako dobimo:

$$D = \begin{bmatrix}
+ & + & + & - & + & - & - \\
- & + & + & + & - & + & - \\
- & - & + & + & + & - & + \\
+ & - & - & + & + & + & - \\
- & + & - & - & + & + & + \\
+ & - & + & - & - & + & + \\
+ & + & - & + & - & - & + \\
- & - & - & - & - & - & -
\end{bmatrix}.$$

Glavni vpliv vsakega dejavnika $X_j$, $j = 1, \ldots, 7$, izračunamo kot razliko povprečij rezultatov na visokem stanju za $Y_j$ $(+)$ in nizkem stanju $Y_j$ $(-)$ po enačbi 4.4 [19].

Konstantni člen $\beta_0 \approx b_0$ pa je definiran kot povprečje vseh rezultatov.

$$E_{X_j} = \mathbb{E}[b_j] = \frac{\sum Y_i(+)}{N/2} - \frac{\sum Y_i(-)}{N/2}, \quad j = 1, \ldots, 7 \tag{4.4}$$
$$E_{X_0} = \mathbb{E}[b_0] = \frac{\sum_{i=1}^{N} Y_i}{N}$$

V tabeli 4.7 smo povzeli matriko modela $X$ v PB načrtu glede na linearni in klasični model. Za oznake posameznih dejavnikov (klasični način) smo vzeli kar oznake iz tabele 4.1. Posebej smo označili število ponovitev Gibbsovega vzorčevalnika z oznako $R$ in funkcijo za izračun psevdo števil s črko $B$. Linearni model prvega reda je razširjena enačba 4.2 na sedem dejavnikov. Ker nas zanimajo samo glavni vplivi, so v PB načrtu vse interakcije med dejavniki zanemarljive.

Tabela 4.7: Prikaz matrike modela glede na linearni model prvega reda in število eksperimentalnih kombinacij. Klasične oznake sovpadajo oznakam dejavnikov v modelu. Rezultati eksperimentalnih kombinacij so simbolično prikazani v zadnjem stolpcu.

| **Model** | | **Vpliv dejavnika** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Linearni** | | $\beta_0$ | $\beta_1$ | $\beta_2$ | $\beta_3$ | $\beta_4$ | $\beta_5$ | $\beta_6$ | $\beta_7$ | $\boldsymbol{Y}$ |
| **Klasični** | $N$ | konst. | $M$ | $W$ | $E$ | $P$ | $T$ | $B$ | $O$ | **rezultat** |
| | 1 | + | + | + | + | − | + | − | − | $Y_1$ |
| | 2 | + | − | + | + | + | − | + | − | $Y_2$ |
| | 3 | + | − | − | + | + | + | − | + | $Y_3$ |
| | 4 | + | + | − | − | + | + | + | − | $Y_4$ |
| | 5 | + | − | + | − | − | + | + | + | $Y_5$ |
| | 6 | + | + | − | + | − | − | + | + | $Y_6$ |
| | 7 | + | + | + | − | + | − | − | + | $Y_7$ |
| | 8 | + | − | − | − | − | − | − | − | $Y_8$ |

Za statistično interpretacijo rezultatov PB načrta uporabimo $t$-test, s katerim določimo mejno vrednost za značilne vrednosti preko relacije

$$t = \frac{|E_{X_j}|}{(SE)_e} \quad \Leftrightarrow \quad t_\alpha, \qquad j = 1, \ldots, 7 \tag{4.5}$$

kjer je $(SE)_e$ standardna napaka vpliva dejavnika $X_j$ in predstavlja eksperimentalno variabilnost načrta. Določimo jo lahko na različne načine, med drugim ocene vmesnih natančnosti merskih napak (ang. intermediate precision estimates), ocena napake z dodatnimi spremenljivkami ali z interakcijami med dejavniki, ocena iz porazdelitve samih vplivov ali ocena napake z določeno mejo natančnosti.

Pri PB načrtu ima vsak dejavnik dve stanji, zato definiramo standardno napako $SE$ kot

$$SE = \sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}.$$

Standardna odklona $s_1$ in $s_2$ ocenjujeta varianco rezultatov algoritma v dveh stanjih in sta izražena z isto varianco, in sicer $s^2$. Ker je število poskusov na visokem in nizkem stanju dejavnika enako, velja $N_1 = N_2 = N/2$. Potem je standardna napaka za vpliv dejavnika enaka

$$(SE)_e = \sqrt{\frac{s^2}{N/2} + \frac{s^2}{N/2}} = \sqrt{\frac{4s^2}{N}} \overset{N=8}{=} \frac{s}{\sqrt{2}},$$

kjer smo mejo za vzorčni standardni odklon $s$ določili kot $s = \sum_{i=1}^{8} s_i = \sum_{i=1}^{8} \frac{Y_i}{10}$ pri desetih ponovitvah posameznega poskusa. Tako smo mejo napake postavili v interval desetih odstotkov od dobljenega rezultata $Y_i$.

Da določimo kritično mejo za pomembnost vpliva dejavnika $X_j$, preoblikujemo relacijo 4.5 v relacijo

$$|E_{X_j}| \quad \Leftrightarrow \quad E_\alpha = t_\alpha \cdot (SE)_e,$$

kjer je $\alpha = 0.05$. Vpliv dejavnika $X_j$ je statistično značilen pri vrednosti $\alpha$, če je $|E_{X_j}| > E_\alpha$.

Vplive ponavadi interpretiramo na grafični način z normalnim verjetnostnim grafom (ang. normal probability plot) ali z pol-normalnim verjetnostnim grafom (ang. half-normal probability plot) [19]. Pri obeh grafičnih predstavitvah vplivov dejavnikov so nepomembni vplivi normalno porazdeljeni okoli ničle in so linearno poravnani skozi ničlo. Pomembni vplivi pa so razpršeni stran od ničle. Pri normalnem verjetnostnem grafu je poleg vrednosti vplivov in razpršenosti vplivov okoli ničle prikazana tudi orientacija vpliva. Pozitivna vrednost vpliva dejavnika $X_j$ pomeni pomembno razliko v rezultatu $Y_i$, kadar določimo za $X_j$ visoko stanje (+). Analogno interpretiramo negativno vrednost vpliva dejavnika $X_j$, samo da je $X_j$ bil v nižjem stanju (−). Pri pol-normalnem verjetnostnem grafu se ta informacija izgubi, saj so analizirane absolutne vrednosti vplivov. Podrobnosti so izven obsega tega dela, vendar si jih lahko bralec pogleda v literaturi [3] in [19].

Z načrti FFD in PB lahko dobimo veliko informacij o dejavnikih modela z minimalnim številom testiranj. Občutljivi so na manjkajoče vrednosti, uporabo napačnih stanj, spremembo stanj, napačnega izračuna rezultata in napake pri meritvah, vendar so tudi primerni pri obdelavi več dejavnikov hkrati in kadar lahko predpostavimo, da ima (vsaj delež) interakcij med dejavniki zanemarljiv vpliv pri dobivanju rezultata. Uporabimo jih lahko tudi kot preliminarno obdelavo dejavnikov v metodi, za kar smo tudi uporabili PB načrt v tem delu. Z načrtom smo naredili preliminarno obdelavo prostih dejavnikov pri algoritmu *GraphGibbs*, da smo lahko določili pomembnost posameznih dejavnikov na rezultat algoritma, kar nam je omogočilo optimizacijo postopka.

### Analiza konvergence

Tehnično podlago za obstoj konvergence in potrebne pogoje smo podali z izrekom 2.3.4 v podpoglavju 2.3.2. Na splošno [14], je doseganje konvergence pri iterativnih postopkih lahko oteženo:

1. kadar naredimo premalo iteracij za prehod v stacionarno verigo in

2. kadar je korelacija med prehodi stanj previsoka (kar pomeni, da ne dobimo dovolj "učinkovitih" vzorcev).

Takšne primere lahko razrešimo na tri načine:

1. nadziramo konvergenco s simuliranjem več verig z različnimi začetnimi točkami,

2. primerjamo variance med verigami (ang. between sequence variation) $Var_B$ in variance znotraj verig (ang. within sequence variation) $Var_W$, dokler ne dosežemo $Var_B \approx Var_W$, in

3. spremenimo algoritem, če je učinkovitost vzorčenja prenizka.

Za zmanjšanje vpliva začetne porazdelitve ponavadi zavržemo člene verige do prve polovice števila iteracij, preden naredimo analizo simulirane verige. Delež zavrženih členov verige imenujemo *obdobje zažiganja* $D$ (ang. burn-in period). Kako veliko je obdobje zažiganja in koliko iteracij $T$ je potrebnih, da dobimo stacionarno verigo (ciljno porazdelitev), še vedno predstavlja problem pri diagnostiki konvergence.

Priljubljen pristop k določanju vrednosti $D$ in $T$ je uporaba statističnih testov za preverjanje hipotez o stacionarnosti verige, ki spadajo pod *diagnostiko konvergence* (ang. convergence diagnostics). S temi testi poskušamo zaznati nestacionarnost verige, vendar pa jih ne moremo uporabiti za potrjevanje staciornarnosti. Zato so ti testi potreben, a ne tudi zadosten pogoj za potrditev stacionarnosti verige. Pogost test za analizo konvergence je *Gelman-Rubinov test* [25], ki smo ga uporabili tudi v tem delu. Postopek je prikazan v algoritmu 8.

Koeficient $R_T$ v Gelman-Rubinovem testu meri ali je število iteracij $T$ zadostno, da dosežemo stacionarno verigo. Predpostavka v testu je, da sta $\widehat{\sigma^2}$ in $Var_W$ asimptotično ekvivalentni. V limiti je potem $R_T \approx 1$, kar nakazuje stacionarno verige. Pri končnem $T$ vrednost $\widehat{\sigma^2}$ preceni pravo varianco $Var_f(H(X))$, medtem ko jo $Var_W$ podceni.

Diagnostični test smo uporabili za analizo delovanja Gibbsovega vzorčevalnika na vseh generiranih množicah. Želeli smo preveriti ali je v primerih, kjer najdemo znane motive, koeficient $R_T \approx 1$. Ko se simulirane verige premešajo oziroma se gibljejo k isti porazdelitvi, potem lahko izračunamo "učinkovito število neodvisnih vzorčenj" kot

$$n_{eff} = MT\frac{\widehat{\sigma^2}}{Var_B}.$$

Število $n_{eff}$ je groba ocena za potrebno število iteracij. Želimo se izogniti trditvi, da je naša simulacija boljša od slučajnega vzorčenja, zato priredimo oceno kot $n_{eff} = \min\{n_{eff}, MT\}$ [14].

**Algoritem 8** Psevdokoda Gelman-Rubinovega postopka za diagnozo konvergence.

**Vhod:** $M$ neodvisnih verig dolžine $T + D$ z različnimi začetnimi točkami.
**Izhod:** Koeficient $R_T$.

1: Zavrži prvih $D$ iteracij v vseh verigah in označi preostanek kot $X_{i1}, \ldots, X_{iT}$, $i = 1, \ldots, M$.
2: Izberi statistiko $H_{it} = H(X_{it})$ in izračunaj:

$$\text{povprečje verige: } \overline{H}_{i*} = \frac{1}{T}\sum_{t=1}^{T} H_{it},$$

$$\text{celotno povprečje: } \overline{H} = \frac{1}{M}\sum_{i=1}^{M} \overline{H}_{i*},$$

$$\text{varianca med verigami: } Var_B = \frac{1}{M}\sum_{i=1}^{M} \left(\overline{H}_{i*} - \overline{H}\right)^2,$$

$$\text{varianca znotraj verige: } Var_W = \frac{1}{MT}\sum_{i=1}^{M}\sum_{t=1}^{T} \left(H_{it} - \overline{H}_{i\cdot}\right)^2,$$

$$\text{aposteriorna varianca: } \widehat{\sigma^2} = \frac{T-1}{T} Var_W + Var_B.$$

3: Za dan $T$ definiraj koeficient $R_T = \frac{\widehat{\sigma^2}}{Var_W}$.

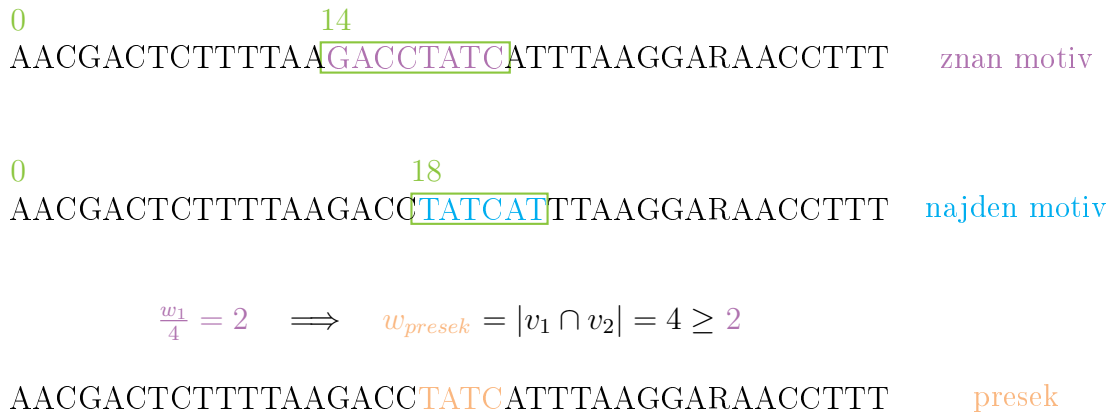
**vrni** koeficiente $R_T$, $Var_B$ in $Var_W$

### 4.2.2 Statistike za merjenje uspešnosti

V vsaki množici poznamo položaje znanih motivov v sekvencah in položaje najdenih motivov. Uspešnost algoritma smo merili glede na vrednosti ocenjevalnih statistik na nukleotidnem nivoju in na vzorčnem nivoju. Na slednjem gledamo, če obstaja presek med položaji najdenih in znanih motivov. Sprejemljiva velikost preseka je vsaj četrtina dolžine znanega motiva. Za zgled si poglejmo sliko 4.2, kjer imamo dva vzorčna niza, znan niz $v_1$ dolžine $w_1 = 8$ in najden motiv $v_2$ dolžine $w_2 = 6$.

Na nukleotidnem nivoju določimo, kako velik je ta presek. S tem lahko definiramo običajne ocene pozitivnih (ang. True Positive) in lažno pozitivnih zadetkov (ang. False Positive), kakor tudi pojem negativnih (ang. True Negative) in lažno negativnih zadetkov (ang. False Negative) [48]:

- $sTP$ - število pojavitev znanih vzorčnih nizov, ki se pokrivajo z najdenimi vzorčnimi nizi,
- $sFN$ - število pojavitev znanih vzorčnih nizov, ki se ne pokrivajo z najdenimi vzorčnimi nizi in
- $sFP$ - število pojavitev najdenih vzorčnih nizov, ki se ne pokrivajo z znanimi vzorčnimi nizi.




Slika 4.2: Primerjava znanega in najdenega vzorčnega niza iste sekvence na vzorčnem nivoju z velikostjo preseka na nukleotidnem nivoju.

Analogno lahko definiramo te pojme na nukleotidnem nivoju:

- $nTP$ - število nukleotidov v preseku znanih in najdenih vzorčnih nizov,
- $nFN$ - število nukleotidov v znanih vzorčnih nizih, ki se ne pokrivajo z najdenimi vzorčnimi nizi,
- $nFP$ - število nukleotidov v najdenih vzorčnih nizih, ki se ne pokrivajo z znaninimi vzorčnimi nizi in
- $nTN$ - število nukleotidov, ki niso del znanih in najdenih vzorčnih nizov.

S temi ocenami lahko na obeh nivojih izračunamo naslednje statistike:

i. občutljivost (ang. SeNsitivity):

$$nSn = nTP/(nTP + nFN) \text{ in} \tag{4.6}$$
$$sSn = sTP/(sTP + sFN), \tag{4.7}$$

ii. pozitivna napovedna vrednost (ang. Positive Predictive Value):

$$nPPV = nTP/(nTP + nFP) \text{ in} \tag{4.8}$$
$$sPPV = sTP/(sTP + sFP). \tag{4.9}$$

Na nukleotidnem nivoju lahko izračunamo še:

iii. specifičnost (ang. SPecificity):

$$nSP = nTN/(nTN + nFP), \tag{4.10}$$

iv. koeficient uspešnosti (ang. Performance Coefficient):

$$nPC = nTP/(nTP + nFN + nFP), \text{ in} \tag{4.11}$$

v. korelacijski koeficient[a] (ang. Correlation Coefficient):

$$nCC = \frac{nTP \cdot nTN - nFN \cdot nFP}{\sqrt{(nTP + nFN)(nTN + nFP)(nTP + nFP)(nTN + nFN)}}. \tag{4.12}$$

Na vzorčnem nivoju lahko izračunamo še povprečna uspešnost (ang. Average Site Performance), to je:

$$sASP = (sSn + sPPV)/2. \tag{4.13}$$

[a]Pearsonov koeficient korelacije za primer dveh spremenljivk

# 5. Rezultati

Delovanje algoritma *GraphGibbs* smo testirali z generiranimi in realnimi množicami DNA sekvenc. Slednje množice smo dobili preko prosto dostopne referenčne baze, ki jo je zgradil Tompa s sodelavci [48]. Na obeh tipih podatkovnih množic smo preverili uspešnost algoritma s statistikami občutljivosti in pozitivne napovedne vrednosti na vzorčnem in tudi nukleotidnem nivoju, kjer smo izmerili še specifičnost algoritma poleg koeficientov uspešnosti in korelacije.

Generirani podatki omogočajo boljšo kontrolo pri testiranju, saj so karakteristike znane vzorčne množice natančno določene. Zato smo dodatno naredili še analizo konvergence na podlagi Gelman-Rubinovega diagnostičnega testa in preverili, kako so množice porazdeljene glede na določanje rešitve in na odstotek ujemanja znanega in najdenega vzorca. Pri teh predstavitvah pogosto uporabimo izraz prvi del algoritma in drugi del algoritma kot ločeni enoti, vendar je drugi del algoritma povezan s prvim delom. Četudi uporabimo samo izraz drugi del, to pomeni, da je algoritem izvedel tako grafični model sekvenc kot Gibbsovo vzorčenje, in ne samo Gibbsovo vzorčenje.

## 5.1 Načrt Plackett-Burman

Z načrtom Plackett-Burman smo preverili vpliv sedmih prostih dejavnikov (nastavitev algoritma) na izhod algoritma *GraphGibbs*. Merili smo vpliv dejavnikov na statistiko $I$ (3.3), s katero ocenimo poravnavo in na odstotek ujemanja $U$ najdenega motiva z znanim motivom. Odstotek ujemanja pomeni delež dolžine podniza v preseku znanega in najdenega motiva izražena v odstotkih.

V tabeli 5.1 so podana visoka (+) in nizka (−) stanja za posamezni dejavnik v PB načrtu. Oznake dejavnikov odgovarjajo oznakam točk v pol-normalnih grafih. Število 0 pomeni, da za ta dejavnik algoritem sam določi optimalno vrednost. V primeru števila motivov to pomeni, da algoritem išče motive, dokler ne najde vseh (statistično nadvpovprečno) zastopanih motivov v množici podatkov. Ker imamo pri vrednosti 0 več možnosti, to predstavlja visoko stanje dejavnika. Nizko stanje pa je določena vrednost, ki jo poda uporabnik. Kadar iščemo motiv dolžine $W = 6$, vstavimo za dolžino motiva uporabniško vrednost 6.

Porazdelitev števila pričakovanih pojavitev motiva $P$ ima na visokem stanju vre-

Tabela 5.1: Visoka in nizka stanja sedmih dejavnikov v Plackett-Burmanovem eksperimentalnem načrtu.

| Dejavnik | Oznaka | Nizko stanje | Visoko stanje |
|---|---|---|---|
| število motivov | $M$ | določeno | 0 |
| dolžina motiva | $W$ | določeno | 0 |
| št. pričakovanih pojavitev motiva | $E$ | določeno | 0 |
| porazdelitev št. pojavitev motiva | $P$ | $nu$ | $u$ |
| število iteracij | $R$ | $\min\{100, 10 \cdot N\}$ | $\max\{100, 10 \cdot N\}$ |
| psevdo-funkcija | $B$ | $\log_{10}(N+1)$ | $\sqrt{N+1}$ |
| orientacija sekvenc | $O$ | $Re$ | $F$ |

dnost $u$, kajti takrat predpostavimo isto število pojavitev motiva na sekvenco, kar v večini primerov pomeni večje število pojavitev najdenega motiva. Pri dejavniku $R$, to je številu iteracij Gibbsovega vzorčevalnika, smo vzeli minimum $\{100, 10 \cdot N\}$ za nizko stanje in maksimum za visoko stanje. Avtorji osnovnega algoritma [27] so predlagali $R = 10 \cdot N$, torej desekratno vrednost števila sekvenc $N$ v dani množici. Pri testnih množicah imamo $N = 5$ in $N = 15$, tako da smo vzeli vrednost 100 kot srednjo vrednost med 50 in 150. S tem smo poenotili razliko med stanjema dejavnika $R$ za vse testne množice.

Dejavnik $B$ predstavlja funkcijo $B$, s katero določimo vrednosti psevdo števil $b_j$ za posamezne črke $r_j \in \mathcal{A}$, $j = 1, \ldots, K$, kjer je $\mathcal{A}$ abeceda trojčkov 3.1 za $K = 64$. Velikost psevdo števil $b_j$ in funkcije $B$ vpliva na izračun elementov pozicijsko uteženе matrike $\mathcal{Q}$, ki jih poračunamo z formulo 3.1. Avtorji osnovnega algoritma so predlagali $B = \sqrt{N}$ za zadovoljive rezultate. Njihov predlog za psevdo-funkcijo sloni na empiričnih poskusih, zato lahko preverimo več različnih možnosti: $\log_{10}(N+1)$, $\sqrt{N/2}$, $\sqrt{N}$ in $\sqrt{N+1}$. Med njimi predstavlja $\log_{10}(N+1)$ najnižjo stanje in $\sqrt{N+1}$ najvišje stanje.

Orientacija sekvence pove, katero verigo dvojne vijačnice beremo. Oznaka $F$ pomeni, da obdelamo dano sekvenco in $Re$ njej odgovarjajočo sekvenco v DNA verigi, ki je zgrajena iz zaporedja komplementarnih nukleotidnih baz, kot je prikazano na sliki 1.4. Zaradi slučajnega elementa v prvem delu algoritma pri določanju desetih točk za induciran podgraf lahko orientacija sekvence, posebno pri variabilnih in kratkih motivih, vpliva na rezultat algoritma *GraphGibbs*. Ti dve stanji sta nominalni, zato je določitev orientacije v posameznem stanju poljubno določena.

Načrt smo izvedli na generiranih testnih množicah po shemi 4.1. V grobem smo razdelili testne množice na tri skupine, ki so ločene glede na dolžino motiva $W \in \{6, 18\}$ in stopnjo variabilnosti $dv \in \{1, 3\}$, pri čemer je $dv = 3$ samo pri dolžini $W = 18$. Množice smo ločili še z ostalimi karakteristikami iz tabele 4.1, to so $P \in \{u, nu\}$, $E \in \{N, 2N\}$, $N \in \{5, 15\}$ in $L \in \{100, 2500\}$. Testne množice smo ozačili po naslednjem zapisu:

$$*\_W\_dv\_P\_E\_N\_L.$$

Zvezdica je generična oznaka, ki predstavlja statistiko $I$ ali odstotek ujemanja $U$.

Namesto zvezdice je lahko tudi oznaka $T1$, ki zaznamuje testno množico z enim motivom.

Najprej pregledamo vplive vseh dejavnikov po posameznih skupinah. Zaradi lažje preglednosti pri grafičnem prikazu smo 16 testnih množic posamezno skupino

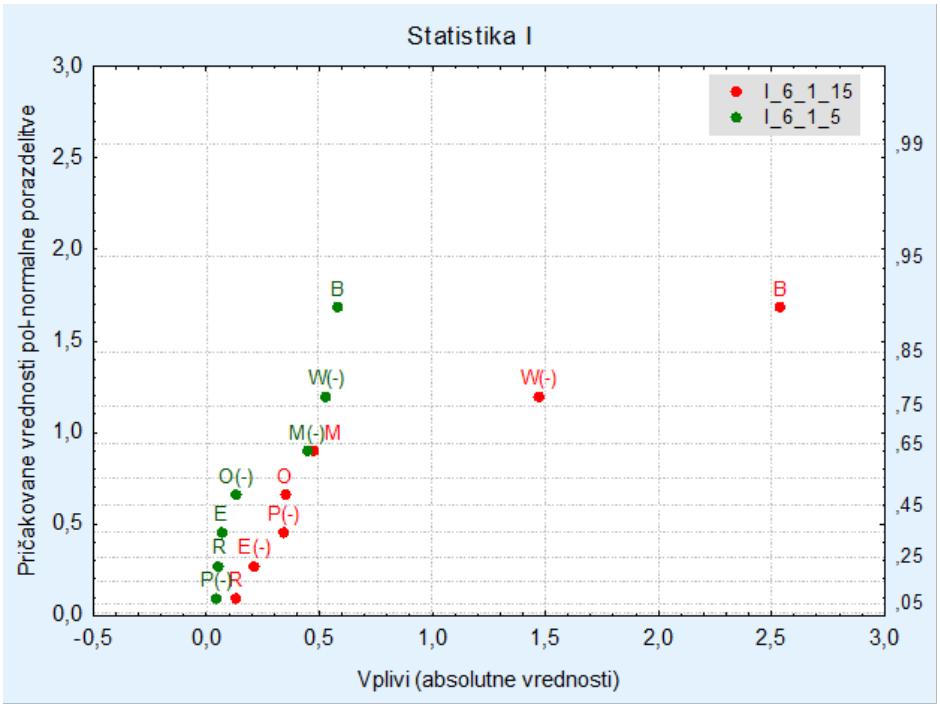
$$*\_W\_dv$$

razdelili še v dve podskupini glede na število sekvenc v množici, torej glede na vrednosti $N = 5$ ali $N = 15$. V posamezni podskupini dobimo tako 8 rezultatov, kar je še vedno precej zgoščeno pri 7 dejavnikih. Zato smo za posamezni poskus v načrtu kot rezultat $Y$ v tabeli 4.7 podali vrednost mediane statistike $I$ ali odstotka ujemanja $U$ po vseh osmih množicah.

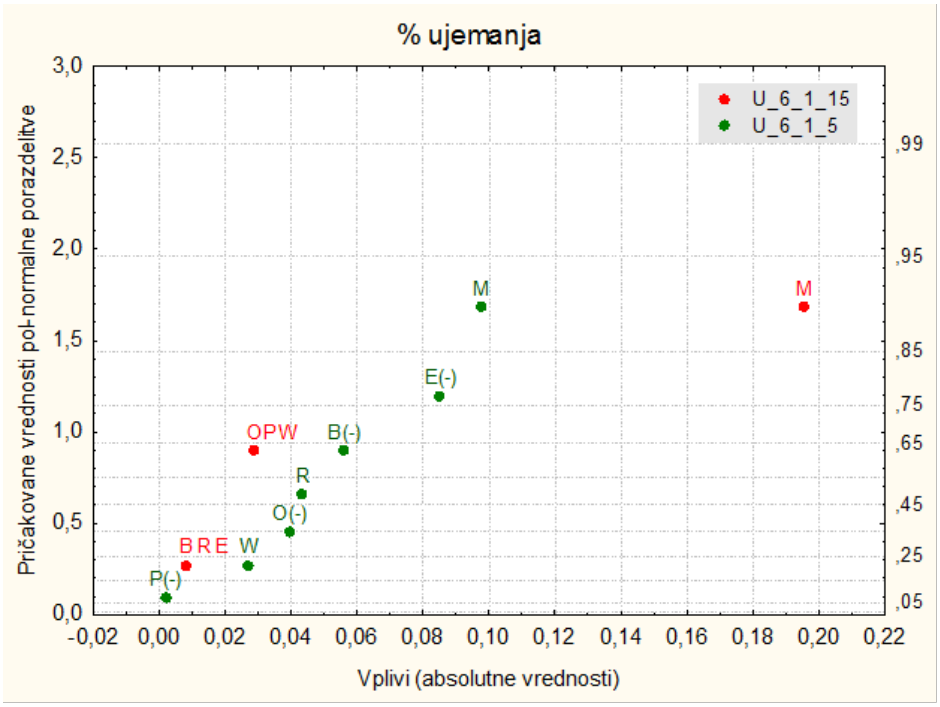
Vplive vseh dejavnikov po skupinah $*\_W\_dv\_N$, kjer je $* \in \{I, U\}$, smo prikazali s pol-normalnim verjetnostnim grafom. Pol-normalna porazdelitev slučajne spremenljivke $Z$ je porazdelitev $Z = |X|$, kjer je $X$ normalno porazdeljena. Na $x$-osi imamo podane absolutne vrednosti vplivov razporejenih od najmanjše do največje. Vrednosti vplivov izračunamo po formuli 4.4, vzamemo njihovo absolutno vrednost in jih razporedimo. Na $y$-osi je prikazana porazdelitev vplivov dejavnikov glede na pričakovane vrednosti v pol-normalni porazdelitvi. Skala na levi strani grafa prikazuje standardni odklon $z$-vrednosti, skala na desni strani pa je odgovarjajoča verjetnostna skala. Linearnost točk nakazuje pol-normalnost in posledično tudi zanemarljiv vpliv dejavnika. Nepomembni vplivi so porazdeljeni okoli ničle, medtem ko so pomembni vplivi opazno odmaknjeni od ničle in odstopajo od "linearno poravnanih" točk.

V pol-normalnih grafih točke označujejo dejavnike. Označene so s kraticami kot v tabeli 5.1. Ob nekaterih oznakah smo dodali še $(-)$, kar označuje orientacijo pričakovane vrednosti iz pol-normalne porazdelitve, ki bi jo lahko razbrali samo iz normalne porazdelitve. Graf pol-normalne porazdelitve smo izbrali predvsem zaradi boljše grafične predstavitve, z oznakami orientacije pa smo dodali dodatno informacijo, ki jo dobimo z normalno porazdelitvijo. Oznaka $(-)$ pove, da ima dejavnik večji vpliv na rezultat algoritma, kadar je v nižjem stanju. Točke dveh skupin razlikujemo po barvi, in sicer smo z zeleno barvo označili skupino $N = 5$ in z rdečo barvo skupino $N = 15$.

Na sliki 5.1 sta prikazana grafa vplivov dejavnikov na statistiko $I$ na zgornjem in odstotek ujemanja $U$ na spodnjem grafu. Pri prvem grafu (a) so točke vidno linearno poravnane pri skupini $I\_6\_1\_5$, kjer ima malo večje odstopanje samo dejavnik $B$ (psevdo-funkcija). Večji vpliv pri dejavniku $B$ je pričakovan, saj funkcija $B$ vpliva na izračun pozicijsko utežene matrike $\mathcal{Q}$, kar lahko razberemo iz formule 3.1. Pri drugi skupini pa od linearno poravnanih točk vidno odstopata dva dejavnika, in sicer dejavnik $W$ (dolžina motiva) in dejavnik $B$. Vpliv slednjega je pričakovano večji, posebej v visokem stanju, kjer je $B = \sqrt{N+1}$. Pri $N = 15$ imamo večjo množico podatkov, torej je več nukleotidov v ozadju nasproti številu nukleotidov v vzorčnih




(a) Vplivi dejavnikov na statistiko $I$.




(b) Vplivi dejavnikov za odstotek ujemanja $U$.

Slika 5.1: Vplivi sedmih dejavnikov algoritma pri skupinah $*\_6\_1\_N$ za $N = 5$ in $N = 15$: (a) pol-normalni graf za statistiko $I$ in (b) pol-normalni graf za odstotek ujemanja $U$.

nizih. Prav tako velja

$$N = 5 : B_{\log} = \log_{10}(5+1) = 0.78 < B_{sqrt+1} = \sqrt{5+1} = 2.45$$
$$N = 15 : B_{\log} = \log_{10}(15+1) = 1.20 < B_{sqrt+1} = \sqrt{15+1} = 4$$

Večji vpliv dejavnika $B$, predvsem v skupini $I\_6\_1\_15$, je pričakovano v visokem stanju.

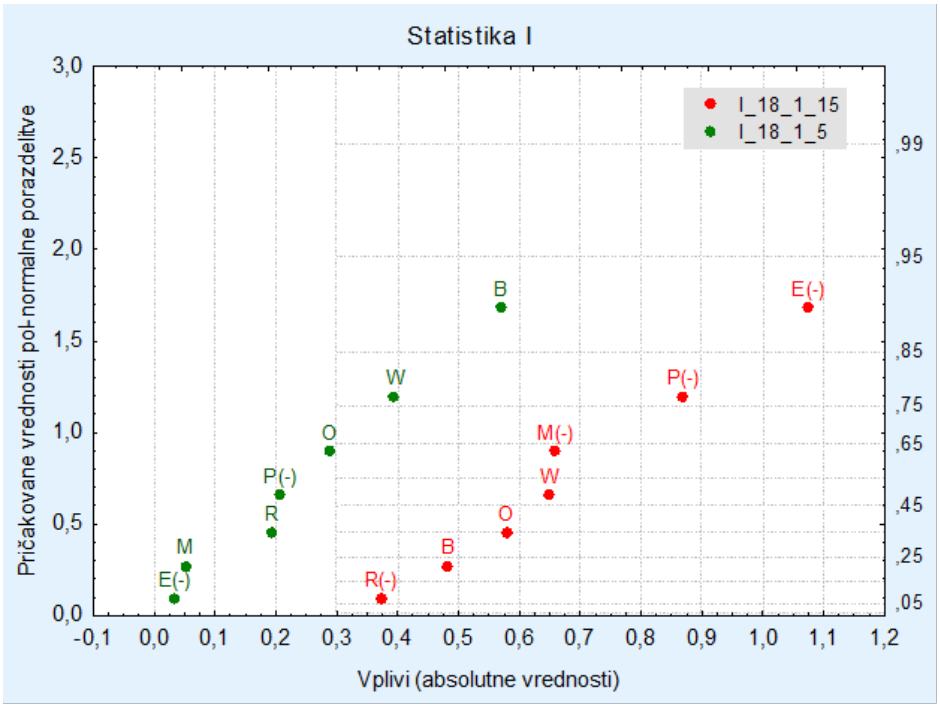
Dodatni vplivni dejavnik pri statistiki $I$ v skupini $I\_6\_1\_15$ je še dolžina motiva $W$ v nizkem stanju, kar pomeni, da je bila dolžina določena s strani uporabnika. Algoritem *GraphGibbs* v prvem delu išče najdaljše popolnoma ohranjene motive, ki imajo vsaj $N/2$ pojavitev v množici. Pri dolžini $W = 6$ lahko najde motive, ki so tudi za več črk daljši od znanega motiva. Dolžina motiva pa še vedno vpliva na velikost statistike $I$, vsaj toliko, da smo to zaznali pri poskusih PB načrta.

Na grafu (b) v sliki 5.1 vidimo, da so vplivi dejavnikov na odstotek ujemanja $U$ v veliki večini zanemarljivi. Pri obeh skupinah ima največji vpliv dejavnik $M$, ki pa pri skupini $U\_6\_1\_15$ vidno odstopa od linearno poravnanih točk. Ker ima $M$ pozitiven predznak, ima dejavnik večji vpliv, kadar algoritem sam poišče število možnih motivov v množici. V nizkem stanju je $M = 1$, saj iščemo samo en (znan) motiv v dani množici. Pri tej nastavitvi pa ni nujno, da je algoritem našel znani motiv. To je posledica omejitev, ki smo jih postavili pri iskanju prvega dela, nastavitev parametrov s strani uporabnika in potrebe po rangiranju ter klasifikaciji vseh najdenih motivov. Tovrstne posledice so pojasnjene v poglavju razprava.

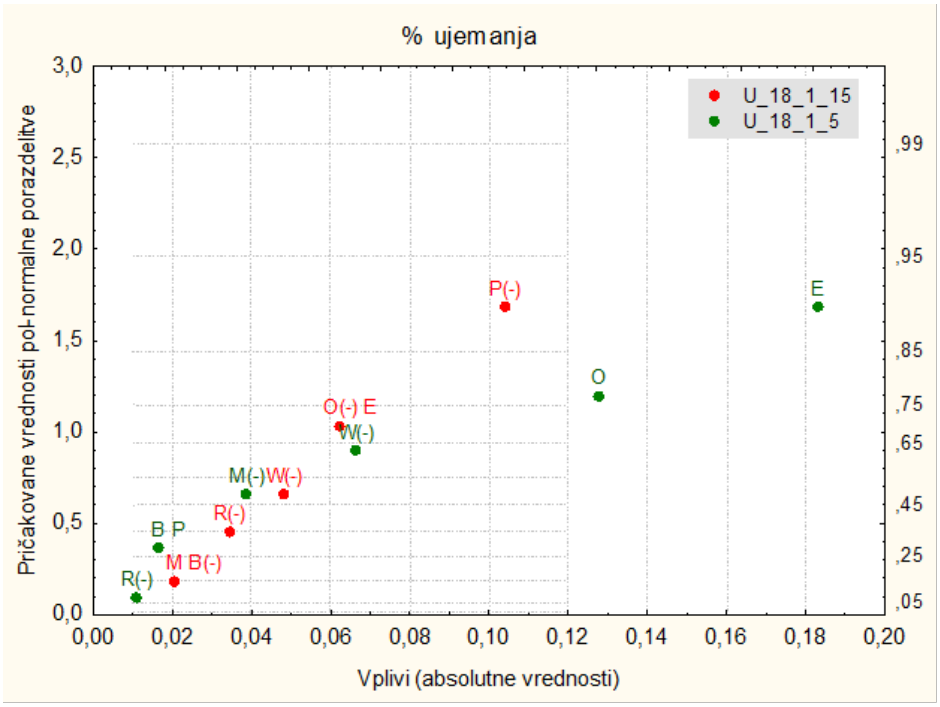
Pri drugi skupini $U\_6\_1\_15$ se točke odgovarjajočih dejavnikov prekrivajo na dveh mestih. Dodatno lahko opazimo, da ima pri obeh grafih (a) in (b) najmanjši vpliv dejavnik $R$, to je število iteracij Gibbsovega vzorčenja, in sicer pri obeh skupinah, kar nakazuje, da pri krajših motivih število iteracij algoritma *GraphGibbs* nima opaznega vpliva na rezultat algoritma.

Pri ostalih skupinah je dolžina motiva $W = 18$, kar je tudi največja dolžina pri generiranih množicah podatkov. Možni sta dve stopnji variabilnosti, in sicer $dv = 1$ in $dv = 3$. Na sliki 5.2 vidimo (a) pol-normalni graf za statistiko $I$ in (b) pol-normalni graf za odstotek ujemanja $U$. Pri obeh grafih lahko zasledimo linearno poravnavo točk pri obeh skupinah, kar nakazuje, da noben dejavnik nima zelo velikega vpliva pri računanju statistike $I$. Pri prvem grafu (a) je za skupino $I\_18\_1\_15$ "linearna poravnava" zamaknjena po $x$-osi, kar nakazuje večje vrednosti vplivov dejavnikov na statistiko $I$. Vzrok je predvsem v večjem številu sekvenc v množici, kar vpliva na velikost statistike $I$. Obe linearni poravnavi, ki jih generirajo točke posamezne skupine, imata približno enak naklon, razlikujeta se predvsem pri rangiranju vplivov dejavnikov. Tako je pri skupini, kjer je $N = 5$, najvišje rangiran dejavnik $B$, kar je pri statistiki $I$ pričakovano.

Drugače je pri skupini $I\_18\_1\_15$, kjer je najvišje rangiran dejavnik število pričakovanih pojavitev motiva $E$, ki ima večji vpliv v nizkem stanju, torej ko smo število pojavitev določili, medtem ko je na primer dejavnik $B$ precej nižje rangiran. To




(a) Vplivi dejavnikov na statistiko $I$.




(b) Vplivi dejavnikov na odstotek ujemanja $U$.

Slika 5.2: Vplivi sedmih dejavnikov algoritma *GraphGibbs* pri skupinah $*\_18\_1\_N$ za $N = 5$ in $N = 15$: (a) pol-normalni graf za statistiko $I$ in (b) pol-normalni graf za odstotek ujemanja $U$.

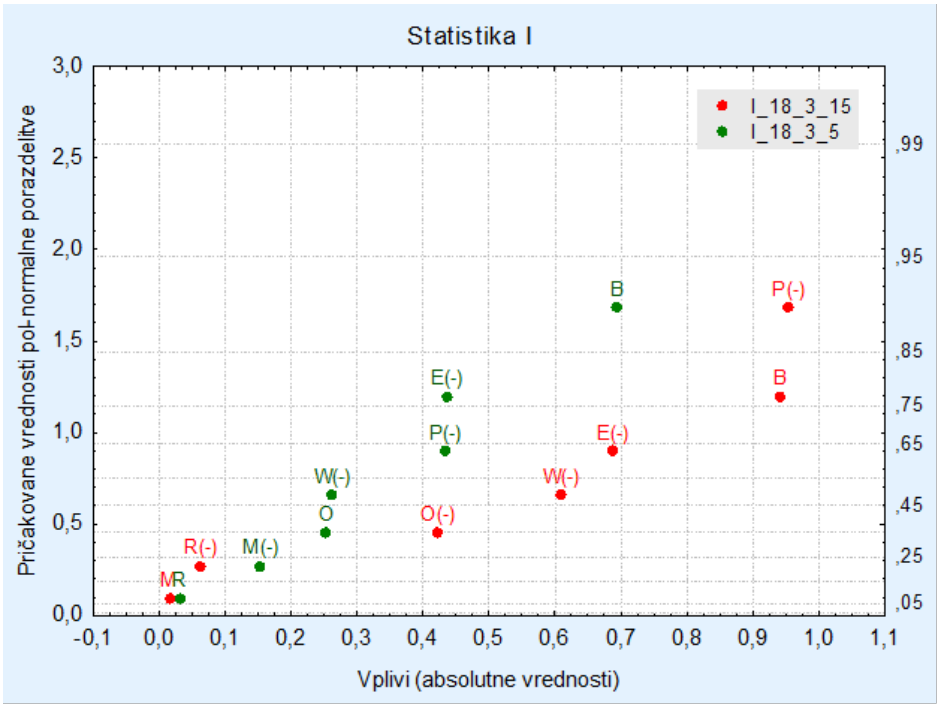
je med drugim posledica dejstva, da je algoritem pri teh testnih množicah pogosto našel vse pojavitve motivov že v prvem delu in se tako Gibbsov vzorčevalnik ni niti zagnal. Lahko pa je nizek vpliv dejavnika $B$ tudi posledica izbire mediane pri združitvi rezultatov osmih množic. Mediana je robustna ocena, ki približa vrednosti posameznih residualov $|E_{X_j}|$, izračunanih po formuli 4.4. Podrobneje je vpliv dejavnika $B$ grafično prikazan na sliki 5.4.

Graf (b) na sliki 5.2 prikazuje vplive dejavnikov na odstotek ujemanja $U$. Podobno kot pri grafu (a) so tudi tukaj vplivi dejavnikov razporejeni po linearni črti pri obeh skupinah. Razlikujeta se predvsem v naklonu. Za skupino $U\_18\_1\_5$ je linearna črta bolj položna in najvišje rangiran dejavnik je dejavnik $E$ na višjem stanju, ki pa ima še vedno zelo majhen vpliv. Ko je dejavnik $E$ na najvišjem stanju, algoritem *GraphGibbs* sam predvidi, kakšno je število pojavitev v prvem delu algoritma, kar vpliva na postopek iskanja, ki smo ga opisali v podpoglavju 3.2.

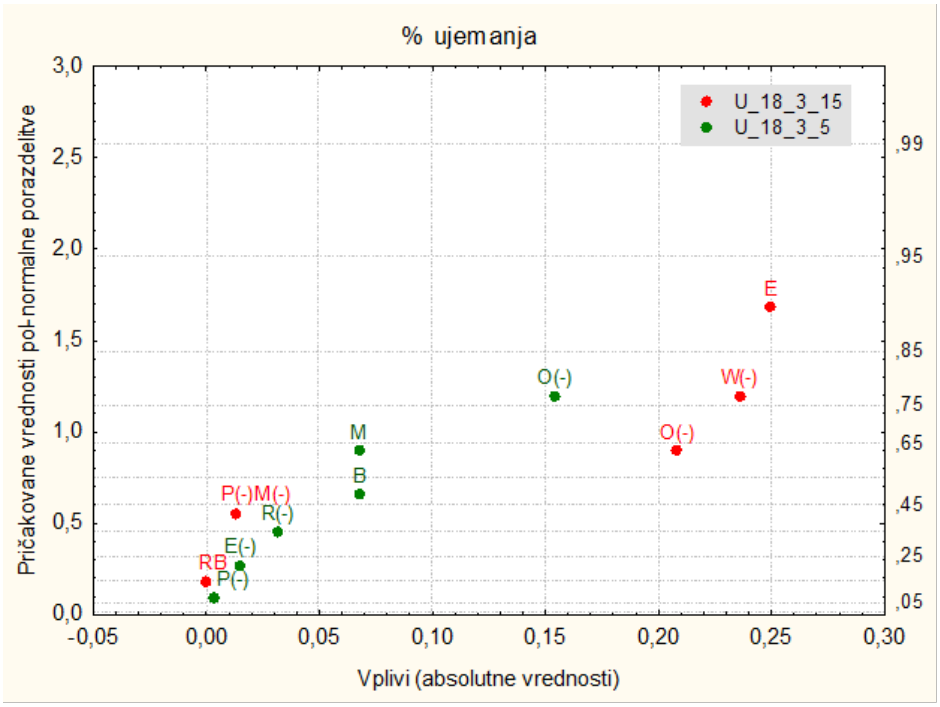
Na sliki 5.3 sta še dva grafa pol-normalne porazdelitve za statistiko $I$ in odstotek ujemanja $U$ za zadnji dve skupini testnih množic, kjer je $W = 18$ in $dv = 3$. Pri skupini $N = 5$ so na obeh grafih (a) in (b) točke dejavnikov linearno razporejene. Na grafu (a) je najvišje rangiran dejavnik $B$, na grafu (b) pa dejavnik $O$, vendar z zelo majhnim vplivom. Na grafu (a) sta pri tej skupini izpostavljena še dejavnika $E$ in $P$ na nizkem stanju, torej vplivata na statistiko $I$, kadar je $E$ določen in algoritem predpostavi neenakomerno porazdelitev vzorčnih nizov.

Kadar število pojavitev motiva ni določeno, lahko algoritem v prvem delu preceni ali podceni število motivov predvsem pri daljših motivih z določeno stopnjo variabilnosti. Variabilnost vzorčnih nizov pomeni kratke podnize, ki so popolnoma ohranjeni v preseku vseh vzorčnih nizov. To lahko vodi do precenjevanja števila vzorčnih nizov, če algoritem najde slučajne pojavitve kratkega podniza v množici. Na drugi strani je lahko presek vseh izbranih vzorčnih nizov samo en trojček, s katerim po postopku algoritma določimo drugo vzorčno množico. Samo število vzorčnih nizov ima tudi manjši vpliv na izračun statistike $I$, s čimer si lahko razložimo tudi rang dejavnika $P$ na nizkem stanju. V visokem stanju algoritem predpostavlja enako število vzorčnih nizov v vsaki sekvenci, kar pri neenakomerno porazdeljenih (znanih) vzorčnih nizih pomeni precenitev števila najdenih vzorčnih nizov.

Pri drugi skupini $I\_18\_3\_15$ grafa (a) na sliki 5.3 najbolj odstopa dejavnik $P$, ki ima največji vpliv na statistiko $I$, ko algoritem predpostavi neenakomerno porazdelitev vzorčnih nizov. Razlog za takšen rang lahko interpretiramo podobno, kot smo pri prvi skupini v predhodnem odstavku. Pri vrednosti $P = u$ algoritem privzame, da je število pojavitev motiva enako v vsaki sekvenci, kjer je število $E$ dano ali pa ga algoritem določi na maksimalno število pojavitev najdenega motiva v prvem delu v posamezni sekvenci. Drugače pa algoritem naredi oceno števila pojavitev motiva s pomočjo funkcije 3.8. Precej visok vpliv ima tudi dejavnik $B$, ki pa je še vedno linearno poravnan z ostalimi rdečimi točkami. Takšen rezultat je bil pričakovan, kar smo pri statistiki $I$ tudi predpostavili pri vseh skupinah. Odstopanje je bilo samo pri skupini $I\_18\_1\_15$.




(a) Vplivi dejavnikov na statistiko $I$.




(b) Vplivi dejavnikov na odstotek ujemanja $U$.

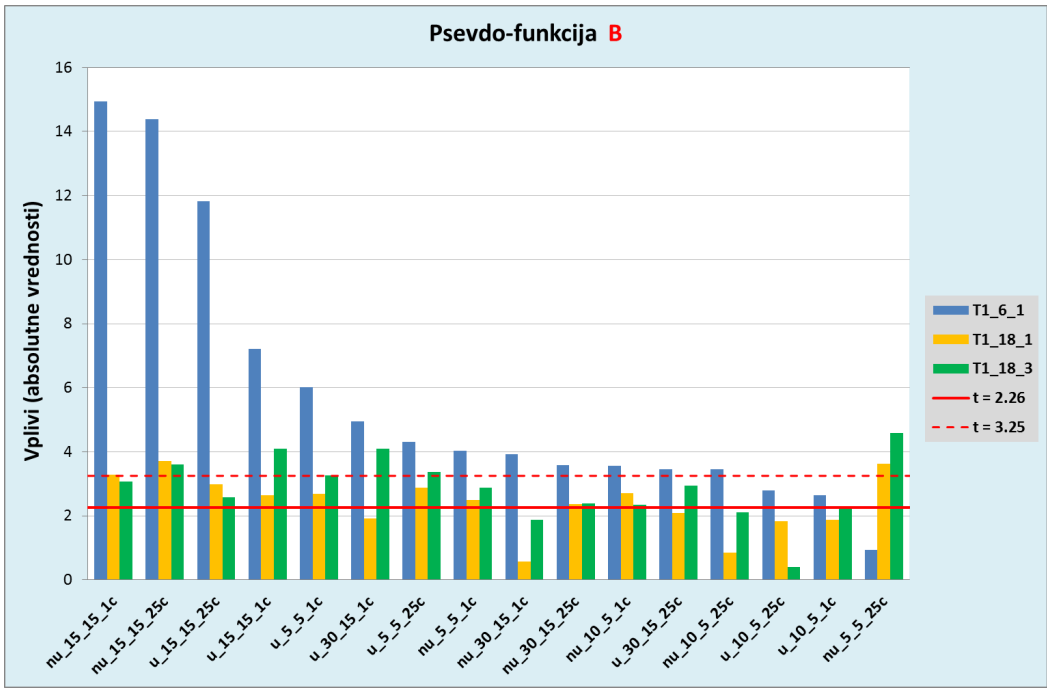
Slika 5.3: Vplivi sedmih dejavnikov algoritma *GraphGibbs* pri skupinah $*\_18\_1\_N$ za $N = 5$ in $N = 15$: (a) pol-normalni graf za statistiko $I$ in (b) pol-normalni graf za odstotek ujemanja $U$.

Graf (b) je predstavljen z manjšo skalo, vendar trije dejavniki vidneje odstopajo od linearno poravnanih točk pri drugi skupini, kjer je $N = 15$. Najvišje rangiran dejavnik je dejavnik $E$, ampak na visoki ravni, kjer algoritem sam določi število vzorčnih nizov. Odstotek ujemanja $U$ je odvisen od števila medsebojno si podobnih nizov, zato lahko določitev števila vzorčnih nizov oblikuje bolj neenotno vzorčno množico, pri kateri dobimo manjši odstotek ujemanja med najdenim vzorcem in znanim motivom. Analogno lahko interpretiramo rang dejavnika $W$ na nizki ravni. Takrat je dolžina določena, tako da algoritem išče motiv dane dolžine ali najdaljši motiv, ki še ima dovolj pojavitev v celotni množici. Dolžina najdenega motiva ima direkten vpliv na izračun odstotka ujemanja $U$; če najdemo krajši motiv od znanega, bo manjši tudi odstotek ujemanja.

Pri pregledu vplivov vseh dejavnikov po različnih skupinah sta bila najpogosteje najvišje rangirana vpliv dejavnika $B$ pri statistiki $I$ in vpliv dejavnika $E$ pri odstotku ujemanja $U$. Zato si vpliv teh dveh dejavnikov oglejmo še podrobneje pri vseh šestnajstih množicah v poddrevesu v shemi 4.1, ki so definirani glede na dolžino motiva in stopnjo variabilnosti. Vsaka skupina je povezana z drugo barvo. Ime celotne množice zapišemo v obliki:

$$\underbrace{T1\_W\_dv}_{\text{določa skupino}}\_\underbrace{P\_E\_N\_L}_{\text{določa množico}},$$

kar se bere kot testna množica z enim znanim motivom $T1$ dolžine $W$ s stopnjo variabilnosti $dv$, kjer je $P$ porazdelitev števila pojavitev $E$ v množici $N$-tih sekvenc z dolžino ozadja sekvence $L$.

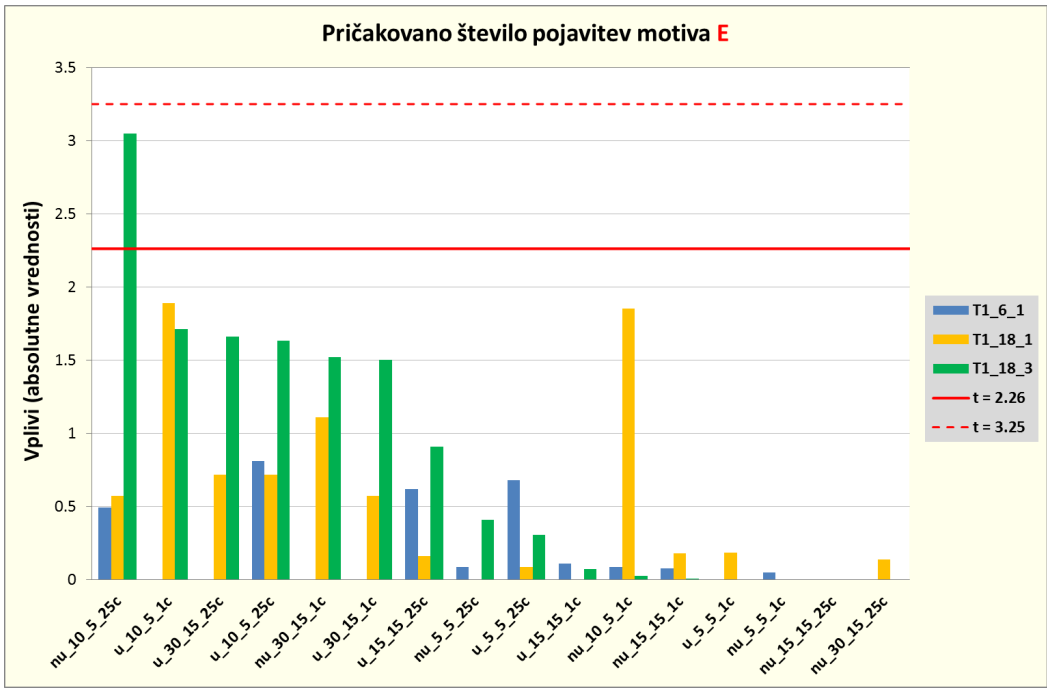



Slika 5.4: Absolutne vrednosti vpliva dejavnika $B$ na statistiko $I$ po množicah, ki jih grupiramo v tri skupine: $T1\_6\_1$, $T1\_18\_1$ in $T1\_18\_3$.

Na sliki 5.4 so s histogrami narisane absolutne vrednosti vplivov dejavnika $B$ na vrednosti statistike $I$ po različnih množicah. Posamezne skupine so ločene z barvami. Dodatno sta na grafu narisani dve rdeči črti. Polna črta predstavlja vrednosti statistike $t$ pri $\alpha = 0.05$, ali dugače $t_{\alpha=0.05} = 2.26$, medtem ko črtkana črta zaznamuje vrednost $t_{\alpha=0.01} = 3.25$. Vrednosti vplivov nad polno rdečo črto so po relaciji 4.5 klasificirani kot pomembni vplivi na izhod algoritma. Dodatno črtkano črto smo dodali za analizo vplivov pri večji stopnji značilnosti $\alpha$.

Vpliv dejavnika $B$ je bil pri večini množic pomemben, predvsem pri skupini $T1\_6\_1$, kar lahko razberemo tudi z grafa 5.1(a). Manjši vpliv je prisoten tudi v skupini $T1\_18\_3$, medtem ko je v skupini $T1\_18\_1$ le pri desetih množicah velikost vpliva prešla polno črto. Povzamemo lahko, da dejavnik $B$ vpliva na vrednost statistike $I$.

Na sliki 5.5 so prikazani vplivi dejavnika $E$ na odstotek ujemanja $U$ po vseh množicah. Kot smo videli na grafih 5.2(b) in 5.3(b) je bil vpliv dejanika $E$ na višji ravni najvišje rangiran med vplivi obeh podskupin, vendar je vpliv majhen, kar lahko razberemo tudi iz grafa 5.5. Vplivi so po vseh skupinah zanemarljivo majhni, razen pri množici $T1\_18\_3\_nu\_10\_5\_25c$. Največji vplivi so pri motivih dolžine $W = 18$ in predvsem, kjer je znano število motivov enako $2 \cdot N$.




Slika 5.5: Absolutne vrednosti vpliva dejavnika $E$ na odstotek ujemanja $U$ po posameznih množicah v vsaki skupini: $T1\_6\_1$, $T1\_18\_1$ in $T1\_18\_3$.

Grafi vplivov ostalih dejavnikov na statistiko $I$ in odstotek ujemanja $U$ so podani v prilogi C.1. Na sliki C.1 sta grafa absolutnih vplivov dejavnika $M$ na rezultat algortima *GraphGibbs*, ki ga merimo s statistiko $I$ in z odstotkom ujemanja $U$. Največji vpliv ima dejavnik $M$ na testne množice v skupini z dolžino motiva $W = 6$, kar smo

že zaznali z pol-normalnima grafoma na sliki 5.1. Za kratke motive in tiste z visoko variabilnostjo je bolje prepustiti algoritmu *GraphGibbs*, da najde vse možne motive ali vsaj večje število motivov. Na sliki C.1 lahko razberemo, da je takšna nastavitev boljša tudi pri množicah z dolgimi sekvencami.

Dolžina motiva ima tudi vpliv na rezultat algoritma, kar vidimo na sliki C.2. Vpliv je zaznan tako pri motivih dolžine $W = 6$ in $W = 18$. Vpliv je bolj izrazit na statistiko $I$ predvsem v množicah z dolgimi sekvencami. Daljše sekvence lahko vsebujejo več motivov, ki so lahko tudi daljši od $W = 6$ in te so v algoritmu preferenčne. Pri zelo dolgih motivih, kjer je višja variabilnost, so popolnoma ohranjeni podnizi, ki jih določi algoritem v prvem delu in so v vseh znanih vzorčnih nizih, krajši. V primeru, da je dolžina motiva nedoločena, algoritem v drugem delu nadaljuje z dolžino popolnoma ohranjenega niza. Kadar pa dolžino določimo, algoritem v drugem delu išče motive z dano dolžino. To ima tudi posledico pri izračunu statistike $I$, še bolj pa pri določanju odstotek ujemanja $U$, kar lahko razberemo z grafa (b) na sliki C.2. Vpliv dejavnika $W$ na odstotek ujemanja $U$ je predvsem opazen v tretji skupini testnih množic, kjer so motivi dolgi $W = 18$ in imajo stopnjo variabilnosti $dv = 3$.

Vpliv dejavnika $E$ na odstotek ujemanja $U$ je podan na sliki 5.5. Iz grafa (a) na sliki C.6 si lahko pogledamo vpliv dejavnika $E$ na statistiko $I$ pri vseh testnih množicah. Vpliv je zaznan predvsem na množicah tretje skupine, kjer obstaja $2 \times N$ vzorčnih nizov motiva. To nakazuje, da na teh testnih množicah, v primeru ko ne določimo števila motivov, algoritem zazna manj vzorčnih nizov, kot jih je dejansko. Število popolnoma ohranjenih vzorčnih nizov, ki jih algoritem zazna v prvem delu, je lahko manjše, kot je znano. Število vzorčnih nizov pa vpliva na izračun pozicijsko utežene matrike $\mathcal{Q}$ kot tudi na izračun vektorja ozadja $\mathcal{P}$, ki je določen z sekvencami brez vzorčne množice. Večanje vzorčne množice ima na ta način manjši vpliv na izračun statistike $I$.

Določitev porazdelitve oziroma dejavnik $P$ ima tudi vpliv na izračun statistike $I$ predvsem na množicah, kjer so vzorčni nizi neenakomerno porazdeljeni, kar lahko razberemo z grafa (a) na sliki C.3. Privzeta vrednosti algoritma je neenakomerna porazdelitev, saj smo jo pogosteje zasledili v primerih realnih podatkov. Pri tej nastavitvi lahko algoritem *GraphGibbs* podceni število vzorčnih nizov, medtem ko pri enakomerni porazdelitvi motiva pogosto pride do precenitve (pravega) števila vzorčnih nizov, kadar so ti neenakomerno porazdeljeni. Kadar nastavimo enakomerno porazdelitev, je potrebno upoštevati še vrednost dejavnika $E$, ki je lahko v poljubnem stanju. Število pojavitev motiva na sekvenco direktno določi število pojavitev za celotno množico, ki pa je lahko precej večja od dejanskega števila. Čeprav smo pri razvijanju metode privzeli, da ima vsaka sekvenca vsaj eno pojavitev motiva in tako tudi pri neenakomerni porazdelitvi določimo v vsaki sekvenci najmanj en niz, je pri slednji ocenitev celotnega števila v manjši okolici pravega števila oziroma je podcenitev ali precenitev števila majhna. Na odstotek ujemanja $U$ je nastavitev porazdelitve motiva zanemarljiva.

Dejavnik $R$, ki predstavlja število iteracij Gibbsovega vzorčevalnika, ima zanemarljiv vpliv na obe meritvi rezultata, torej vpliv na statistiko $I$, ki je prikazan na grafu

C.4 (a) in vpliv na odstotek ujemanja $U$, ki ga razberemo z grafa C.4 (b). Najnižje število iteracij je bilo v primeru množic z $N = 5$ sekvencami, kjer smo določili $R = 10 \cdot N = 50$. Majhna številka iteracij in zanemarljiv glavni vpliv dejavnika $R$ nakazuje hitro konvergenco Gibbsovega vzorčevalnika, kadar se je ta dejansko zagnal.

Orientacija branja sekvenc ima manjši vpliv na statistiko $I$, graf (a) na sliki C.5, predvsem pri prvi skupini testnih množic, kjer je dolžina motiva enaka $W = 6$. Brez podrobnejše analize lahko takšen vpliv pripišemo slučajnemu elementu pri izberi desetih vozlišč v prvem delu algoritma. Izbira enakovrednih vozlišč, torej z isto valenco, je slučajna in je zato pri branju zrcalnih sekvenc spremenjeno zaporedje vozlišč. Posledico takšnega izbora obravnavamo še v podpoglavju 6. Vpliv dejavnika $O$ je bil predvsem na statistiko $I$, medtem ko je pri odstotku ujemanja $U$ zanemarljiv.

## 5.2 Uspešnost algoritma *GraphGibbs*

Glede na rezultate PB načrta, ki smo jih analizirali v predhodnem poglavju, smo nekaj parametrov fiksirali, preden smo preverili uspešnost algoritma tako na generiranih kot na realnih množicah DNA sekvenc. Edini nedoločen parameter pri testiranju generiranih in realnih množic je dejavnik $M$, to je število različnih motivov v množici. Parametra $W$ in $E$ sta bila določena glede na znano število vzorčnih nizov in dolžino motiva v generirani množici. Njuni vrednosti tako odgovarjata karakteristiki motiva v posamezni generirani množici. Vrednosti ostalih parametrov smo določili za vse množice enako, in sicer:

- $B = \log_{10}(N + 1)$,
- $R = 250$,
- $O = F$, in
- $P = nu$.

Dejavnika $P$ in $O$ sta imela zanemarljiv vpliv na odstotek ujemanja $U$. Iz polnormalnih grafov lahko vidimo, da je vpliv dejavnika $P$ predvsem na statistiko $I$ večji, ko je $P$ v nizkem stanju. Pri dejavniku $O$ je vpliv posameznega stanja izrazitejši pri različnih skupinah, tako da smo na koncu določili branje kar originalne sekvence, kajti za natančnejšo interpretacijo vpliva dejavnika $O$ v različnih stanjih so potrebne podrobnejše analize.

Izbira psevdo-funkcije ima očiten vpliv na statistiko $I$, a zanemarljiv vpliv na odstotek ujemanja $U$, kar vidimo v grafu (b) na sliki C.6. S pomočjo funkcije $B$, ki določi vrednosti psevdo števil, utežimo matriko $\mathcal{Q}$. Višje kot so vrednosti psevdoštevil, večje bodo dane uteži. Glavna naloga psevdo števil je, da odstranijo možnost neničelne frekvence črke oziroma trojčka v sekvenci. Potrebno je torej določiti majhno pozitivno vrednost posamezni črki abecede, tako da nobena črka nima ničelne frekvence v množici. Ker nadomeščamo ničlo, potrebujemo majhno število, zato smo za psevdo-funkcijo izbrali logaritemsko funkcijo. Ta ima direkten vpliv na velikost statistike $I$, ampak ne tudi na primerjanje vrednosti statistik $I$, ki jih poračunamo

na podlagi enako uteženih matrikah $\mathcal{Q}$ posameznih testnih množic.

Število iteracij $R$ smo fiksirali, saj je bil vpliv tega dejavnika po PB načrtu zanemarljiv povsod pri odstotku ujemanja $U$ in z nekaj izjemami tudi pri statistiki $I$, kar lahko vidimo tudi iz slike C.4. Pri PB načrtu smo za dejavnik $R$ izbirali med tremi možnimi stanji, in sicer 50, 100 in 150, saj so bile vrednosti odvisne od števila sekvenc v testni množici. Ker smo zaznali vpliv dejavnika $R$ na statistiko $I$, smo se odločili povišati število iteracij na 250, kar je vsota vrednosti 150 (najvišje stanje) in 100 (stanje pri vseh testnih množicah).

Kakovost rezultatov algoritma *GraphGibbs* merimo z statistikami občutljivosti, pozitivne napovedne vrednosti, koeficientom uspešnosti in korelacijskim koeficientom. Vrednost statistik računamo na dveh nivojih, in sicer na nukleotidnem nivoju in vzorčnem nivoju. Formule za izračun so podane v podpoglavju 4.2.2. Za boljši pregled primerjalnih grafikonov nivoja dodatno ločimo po barvi ozadja, kjer svetlo rdeča predstavlja nukleotidni nivo in svetlo zelena predstavlja vzorčni nivo.

Na posamezni množici podatkov smo zaradi preverjanja ponovljivosti algoritem pognali desetkrat. Za predstavitev smo nato vzeli povprečno vrednost parametrov $sTP$, $sFP$, $sFN$, $nTP$, $nFP$, $nFN$ in $nTN$ vseh desetih ponovitev oziroma tistih ponovitev, kjer je algoritem zaznal znani motiv. S temi vrednostmi parametrov smo nato izračunali občutljivost in pozitivno napovedno vrednost algoritma *GraphGibbs* na vzorčnem in nukleotidnem nivojih poleg specifičnosti, koeficienta uspešnosti in korelacijskega koeficienta na nukleotidnem nivoju. Zaradi velikega števila množic smo za vizualne predstavitve množice grupirali v različne skupine in uporabili povprečno uspešnost algoritma na tej skupini, pri čemer smo uporabili uteženo povprečje posameznih statistik po vrednostih vseh množic v skupini.

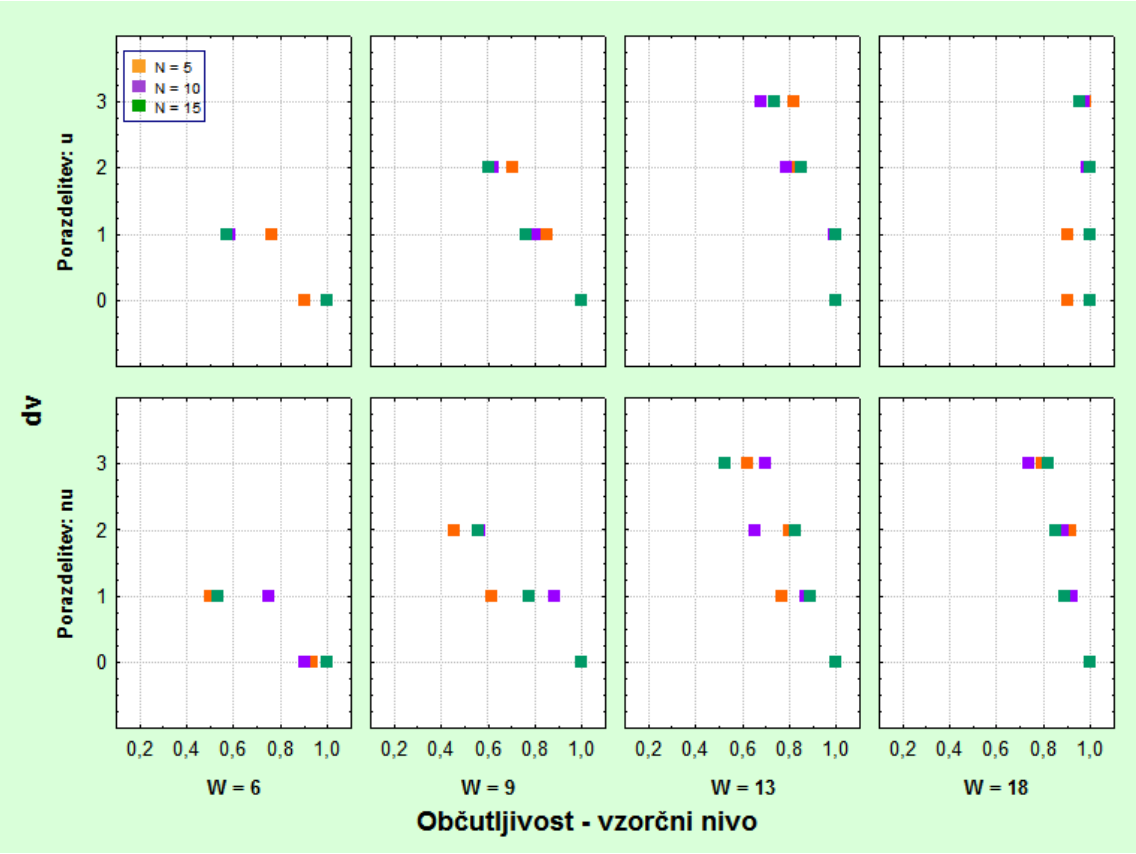
Skupine pri realnih podatkih se primarno ločujejo po vrsti organizma, nato pa še po nastavitvi parametrov algoritma. Za primerjavo statistik upešnosti med skupinami smo uporabili polarne grafikone. Pri generiranih podatkih pa smo uporabili primerjalne kombinatorne grafikone, s katerimi lahko analiziramo razliko med skupinami generiranih množic. Prva ločnica med skupinami je dolžina motiva $W$, nato jih ločimo še glede na stopnjo variabilnosti $dv$. Dodatno jih razlikujemo še glede na število sekvenc $N$ in po porazdelitvi pojavitev znanega motiva $P$ v množici.

### 5.2.1 Generirani podatki

Za prikaz rezultatov v grafikonu smo rezultate posameznih množic zbrali v nekaj večjih skupin, ki se ločijo glede na dolžino motiva $W$, stopnjo variabilnosti $dv$, porazdelitev pojavitev motiva $P$ in število sekvenc $N$ v množici. Točka v primerjalnem grafu prikazuje uteženo povprečje v posamezni skupini in je obarvana glede na število sekvenc v množicah skupine. Tako so množice s petimi sekvencami obarvane z oranžno barvo, množice z desetimi sekvencami so obarvane z vijolično barvo in množice s petnajstimi sekvencami s temno zeleno barvo.

Na sliki 5.6 so prikazani grafikoni, ki prikazujejo občutljivost rezultatov na vzorčnem

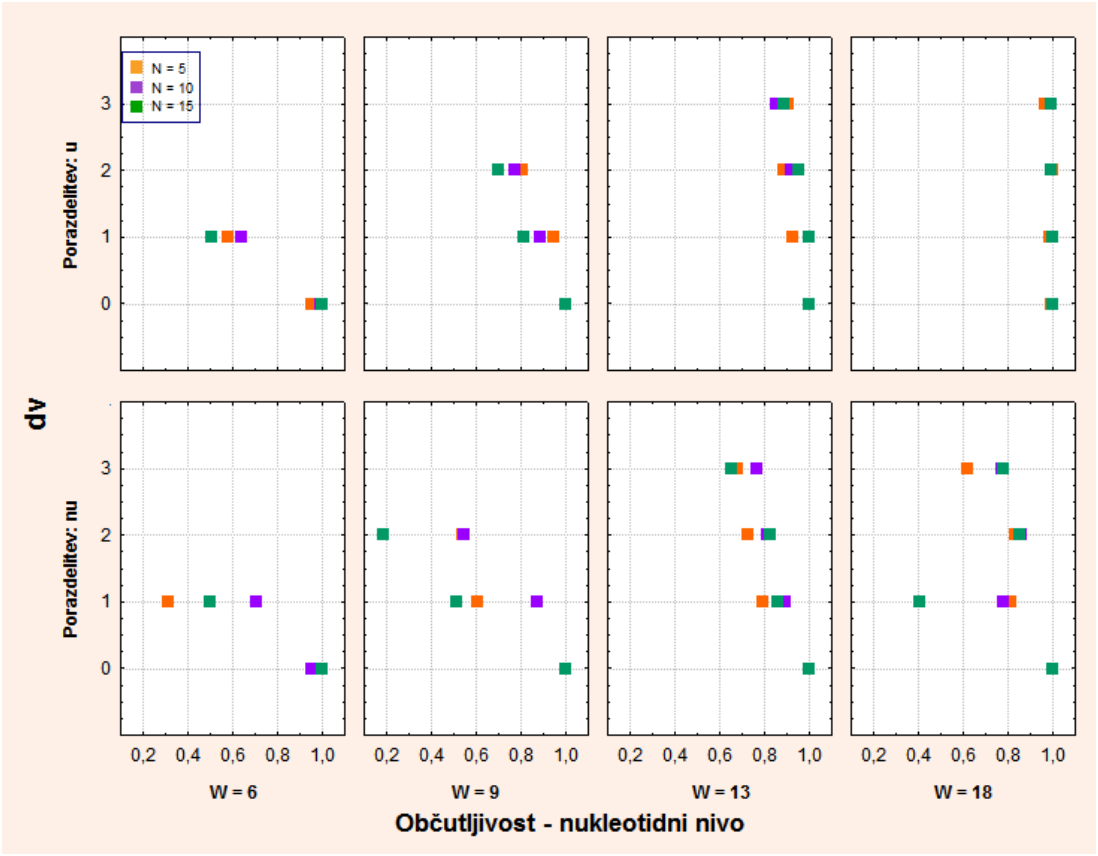
nivoju. Na abscisi imamo podano stopnjo občutljivosti in na ordinati stopnjo variabilnosti. Grafikoni so dodatno razdeljeni v dve skupini na podlagi parametra $P$. Občutljivost rezultata prikazuje razmerje med pravimi zadetki nasproti vseh znanih pojavitev. S to statistiko računamo velikost deleža pravih zadetkov, torej kakšen delež znanih pojavitev motiva je algoritem pravilno zaznal. Na nukleotidnem nivoju računamo velikost preseka med znanimi in najdenimi vzorčnimi nizi. Zaželjeno je torej, da je vrednost te statistike blizu 1.




Slika 5.6: Primerjalni grafi za prikaz občutljivosti algoritma *GraphGibbs* na vzorčnem nivoju med dolžinami motiva $W \in \{6, 9, 13, 18\}$ glede na stopnjo variabilnosti $dv \in \{0, 1, 2, 3\}$, ki so ločeni glede na porazdelitev vzorčnih nizov $P \in \{u, nu\}$.

Na vseh grafikonih na sliki 5.6 so točke linearno poravnane z majhnim negativnim naklonom in z začetno točko v okolici točke $(1, 0)$. To pomeni, da je algoritem zelo občutljiv za zaznavanje popolnoma ohranjenih motivov in za zaznavanje pojavitev daljših motivov, kot na primer $W = 13$ ter $W = 18$, tudi pri višjih stopnjah variabilnosti. Večja odstopanja se pričakovano zgodijo v skupinah s krajšima dolžinama motivov in pri višji stopnji variabilnosti, kjer občutljivost algoritma na nekaterih množicah pade tudi pod mejo 0.5. Na vzorčnem nivoju ni prevelikega odstopanja pri posameznih dolžinah motiva med skupinama množic, ki jih ločimo glede na porazdelitev pojavitev motiva. Razpšenost je za spoznanje večja pri neenakomerni porazdelitvi.

Na vzorčnem nivoju algoritem *GraphGibbs* pretežno zazna velik delež znanih pojavitev motiva med najdenimi pojavitvami. Zmanjšanje občutljivosti pri krajših dolžinah motivov s pozitivno stopnjo variabilnosti je pričakovano, saj je težje najti ohranjen presečni podniz pri kratkih in variabilnih vzorčnih nizih, ki ga iščemo v prvem delu. Ampak upad ni občuten, saj algoritem še vedno zazna znane motive v večini generiranih množic. Prav tako ni opaziti posebnih odstopanj med množicami z različnim številom sekvenc. Ta so bolj izrazita na nukleotidnem nivoju, kar lahko razberemo na sliki 5.7.

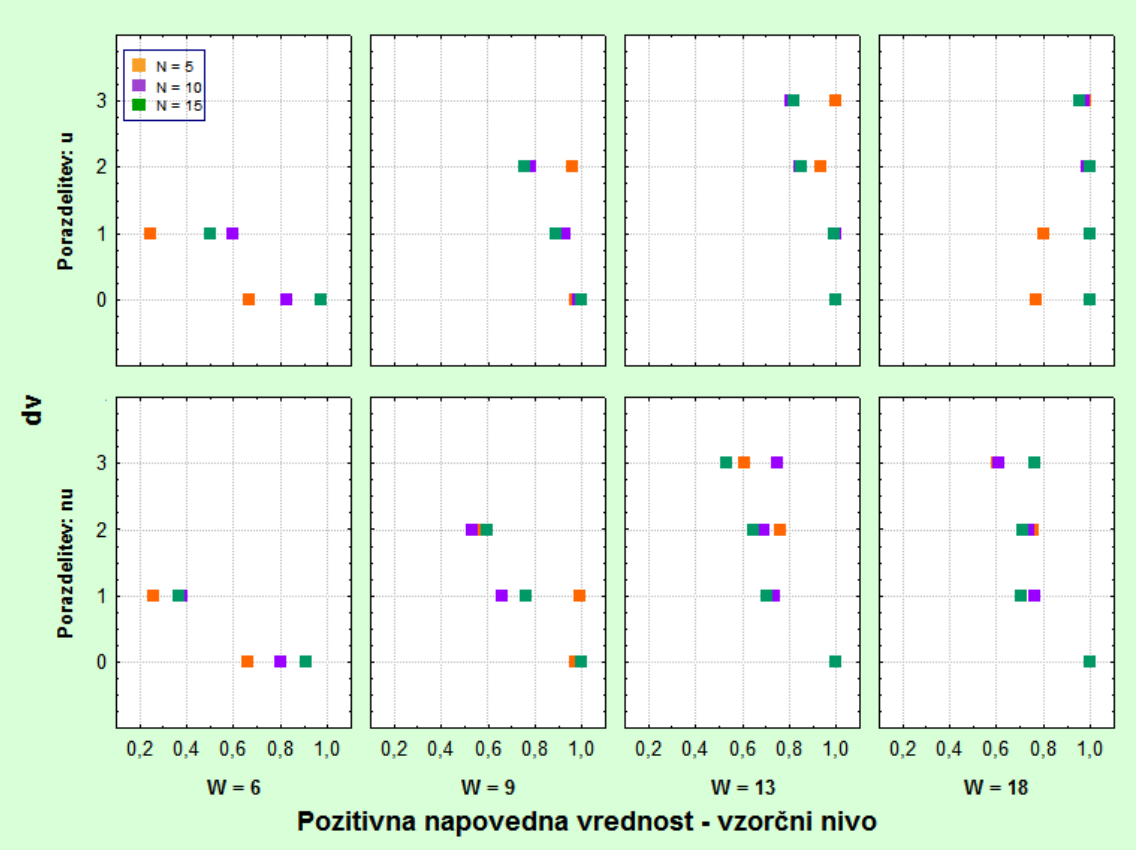



Slika 5.7: Primerjalni grafi za prikaz občutljivosti algoritma *GraphGibbs* na nukleotidnem nivoju med dolžinami motiva $W \in \{6, 9, 13, 18\}$ glede na stopnjo variabilnosti $dv \in \{0, 1, 2, 3\}$, ki so ločeni glede na porazdelitev vzorčnih nizov $P \in \{u, nu\}$.

Na vseh grafikonih na sliki 5.7 je pri $dv = 0$ občutljivost blizu 1 oziroma kar enaka 1. To pomeni, da algoritem *GraphGibbs* zelo dobro zazna popolnoma ohranjene motive. Vrednost statistike je bolj variabilna na višjih stopnjah variabilnosti motiva, posebno pri *nu* porazdelitvi pojavitev znanega motiva. Podobno kot pri sliki 5.6 so točke na vseh grafikonih linearno poravnane z negativnim naklonom. Najbolj je to opazno pri dolžini motiva $W = 13$. Najmanj pa pri krajših dolžinah $W = 6$ in $W = 9$ predvsem pri porazdelitvi *nu*. Pri teh dveh dolžinah ima občutljivost najnižjo vrednost, z izjemo pri množicah z motivom dolžine $W = 18$ in s porazdelitvijo *nu*.

Najboljši rezultati so pri dolžini motiva $W = 18$ in porazdelitvi $u$, kar je ravno posledica teh dveh nastavitev. S slike 5.7 lahko razberemo, da so vrednosti statistike bolj enakomerno porazdeljene, kar nakazuje, da množice z takšnimi porazdelivami bolj odgovarjajo našemu algoritmu. Prav tako lahko razberemo, da je najmanj variabilna vrednost statistike pri skupinah, kjer je $N = 10$ in v skupinah z dolžino motiva $W = 13$.

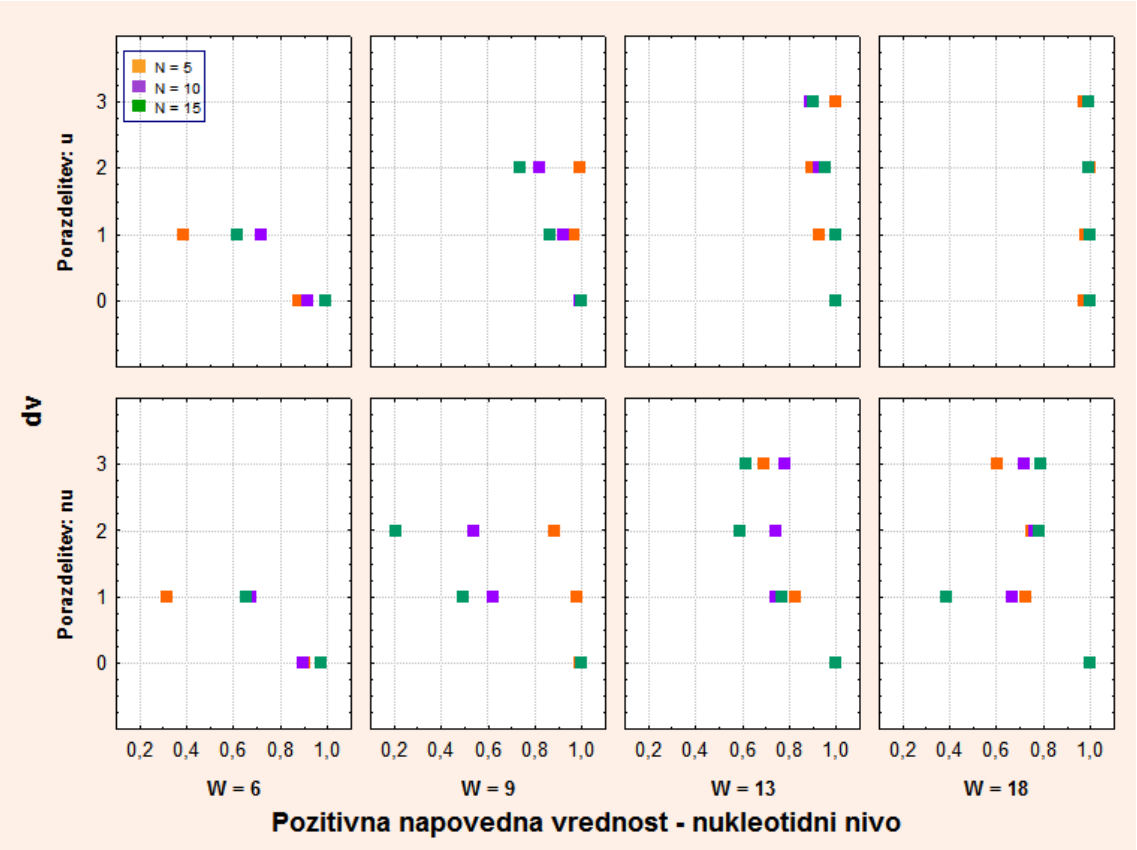
Na obeh nivojih si lahko ogledamo še pozitivno napovedno vrednosti algoritma *GraphGibbs*. Slika 5.8 prikazuje pozitivne napovedne vrednosti algoritma po posameznih skupinah. Pozitivna napovedna vrednost predstavlja delež pozitivnih zadetkov oziroma znanih pojavitev motiva v rešitvi algoritma med vsemi pojavitvami najdenega motiva. Podobno kot pri občutljivosti želimo, da je ta delež čim bližje 1. Na vzorčnem nivoju so grafi precej podobni grafom za občutljivost na sliki 5.6. Nižje vrednosti pozitivne napovedne vrednosti smo dobili predvsem pri motivih dolžine $W = 6$ in $dv = 1$. Najnižja med njimi je pri obeh vrstah porazdelitve pojavitev motiva pri skupini s petimi sekvencami, kar nakazuje, da je v teh primerih algoritem podal več pojavitev, kot jih je dejansko v množici. Vzroki za takšen dogodek so opisani v poglavju 6.




Slika 5.8: Primerjalni grafi za prikaz pozitivne napovedne vrednosti algoritma *GraphGibbs* na vzorčnem nivoju med dolžinami motiva $W \in \{6, 9, 13, 18\}$ glede na stopnjo variabilnosti $dv \in \{0, 1, 2, 3\}$, ki so ločeni glede na porazdelitev vzorčnih nizov $P \in \{u, nu\}$.

Na nukleotidnem nivoju so grafi za pozitivno napovedno vrednost podobni odgovarjajočim grafom na vzorčni ravni. Torej je presek med znanimi in istoležnimi najdenimi vzorčnimi nizi velik, predvsem pri skupinah z daljšim motivom. Pri motivih dolžine $W = 6$ je vrednost PPV na nukleotidnem nivoju precej boljša pri obeh vrstah porazdelitve pri vseh skupinah razen skupine, kjer je $N = 5$. Obratno je pa pri skupinah z motivom dolžine $W = 9$ pri neenakomerni porazdelitvi, kjer je PPV slabša pri množicah s petnajst sekvencami. Na tem grafu lahko tudi vidimo, da je PPV za množice s petimi sekvencami na nukleotidnem nivoju višja kot na vzorčnem nivoju. To pomeni, da je algoritem znane najdene motive zaznal skoraj v celoti.

Na nukleotidnem nivoju je PPV algoritma najboljša na množicah z motivi dolžine $W = 18$ in $W = 13$ pri enakomerni porazdelitvi. Dodatno lahko opazimo visoke vrednosti pri teh množicah za vse stopnje variabilnosti. Tako algoritem tudi v primeru višje variabilnosti motiva zazna najdene znane vzorčne nize skoraj v celoti.

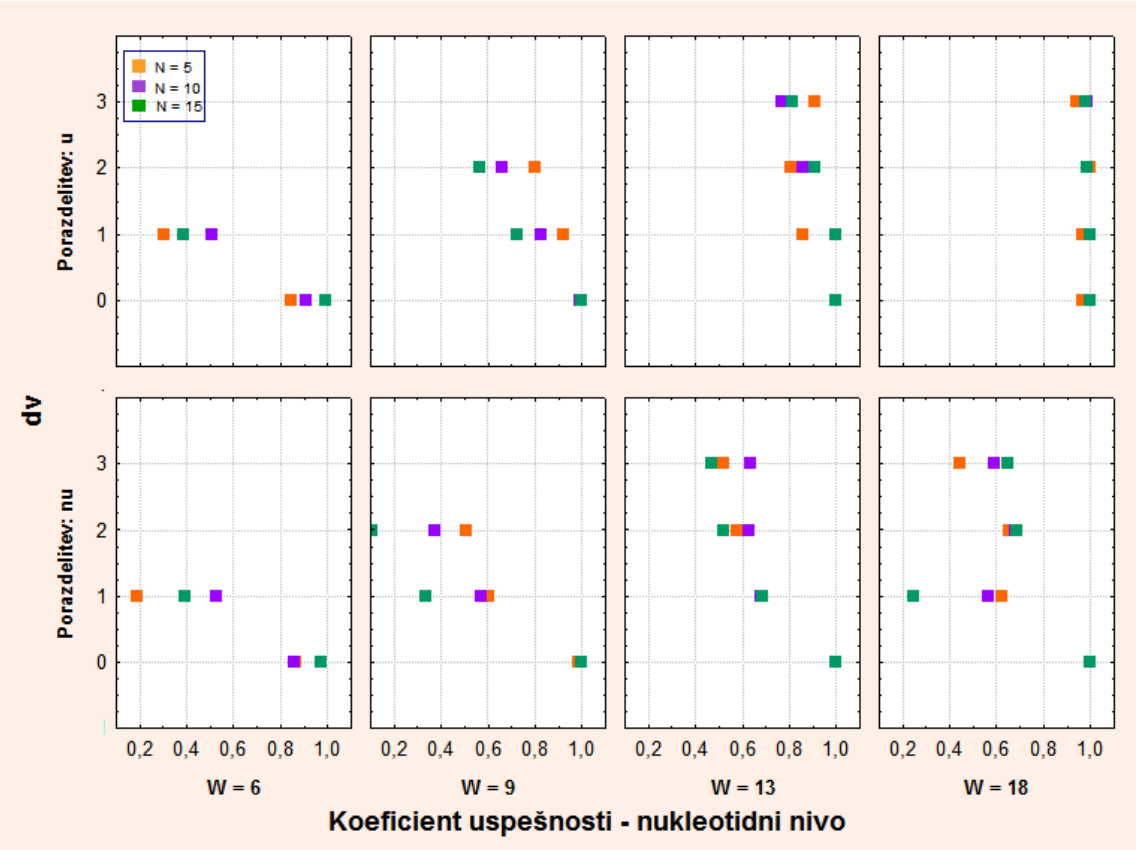



Slika 5.9: Primerjalni grafi za prikaz pozitivne napovedne vrednosti algoritma *GraphGibbs* na nukleotidnem nivoju med dolžinami motiva $W \in \{6, 9, 13, 18\}$ glede na stopnjo variabilnosti $dv \in \{0, 1, 2, 3\}$, ki so ločeni glede na porazdelitev vzorčnih nizov $P \in \{u, nu\}$.

Na nukleotidnem nivoju smo merili vrednosti dveh dodatnih statistik, in sicer koeficient uspešnosti, ki je definiran s formulo 4.11, in korelacijski koeficient, ki ga izračunamo s formulo 4.12. Prva statistika meri uspešnost algoritma kot razmerje med številom najdenih znanih pojavitev s številom vseh možnih pojavitev (najdenih

in znanih). Kadar se najdeni in znani vzorčni nizi prekrivajo, sta ti dve števili enaki in je koeficient enak 1. Korelacijski koeficient je Pearsonov koeficient korelacije, ki ima razpon med vrednostima $-1$ (popolna negativna povezanost) in 1 (popolna pozitivna povezanost). Dva vektorja poravnave, to je znana poravnava motiva in poravnava algoritma, sta popolno pozitivno povezana, ko se popolnoma prekrivata.

Najboljši koeficient uspešnosti smo dobili pri skupini z dolžino motiva $W = 18$ in z enakomerno porazdelitvijo pojavitev motiva, kar vidimo na primerjalnem grafu zgoraj skrajno desno na sliki 5.10. Koeficient je blizu ena za vse množice z različnim številom sekvenc in pri vseh stopnjah variabilnosti. Dobri rezultati so tudi pri množicah z dolžino motiva $W = 13$ in $P = u$. Pri neenakomerni porazdelitvi vrednosti koeficienta uspešnosti pri višjih stopnjah variabilnosti upadejo. Pri vseh podanih skupinah na sliki 5.10 je vrednost blizu oziroma enaka ena za množice s popolnoma ohranjenimi motivi in za vse možne vrednosti $N$.



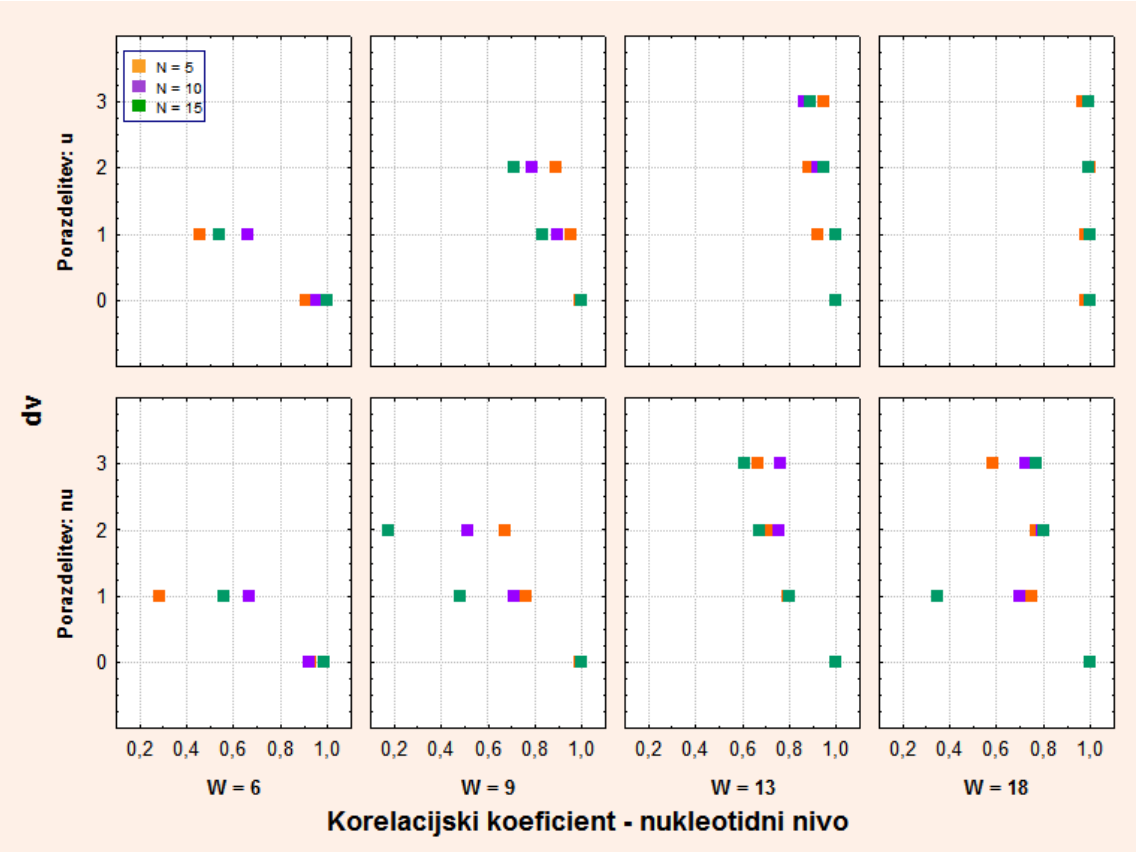

Slika 5.10: Primerjalni grafi za prikaz koeficienta uspešnosti algoritma *GraphGibbs* na nukleotidnem nivoju med dolžinami motiva $W \in \{6, 9, 13, 18\}$ glede na stopnjo variabilnosti $dv \in \{0, 1, 2, 3\}$, ki so ločeni glede na porazdelitev vzorčnih nizov $P \in \{u, nu\}$.

Pri pozitivnih stopnjah variabilnosti koeficient uspešnosti pade tudi blizu ničle, predvsem pri dveh krajših dolžinah in neenakomerni porazdelitvi pojavitev motiva. Pri dolžini $W = 6$ imajo najnižji koeficient uspešnosti množice s petimi sekvencami,

medtem ko imajo množice z motivom dolžine $W = 9$ in s petnajst sekvencami najnižjo vrednost. Večja razpršenost vrednosti koeficienta uspešnosti je pri množicah s porazdelitvijo *nu*, kar vidimo na grafikonih v spodnji vrsti na sliki 5.10.

Oblike primerjalnih grafov na sliki 5.11 za korelacijski koeficient so zelo podobne oblikam primerjalnih grafov na sliki 5.10 za koeficient uspešnosti. Točke za iste skupine množic so pri grafih korelacijskega koeficienta manj razpršene kot pri grafih koeficienta uspešnosti. Oblika grafa in obarvanost točk pa je na dveh slikah enaka. Kjer je korelacija visoka, znana in najdena poravnava sovpadata skoraj v celoti.

V prilogi C.2 je podana slika C.7 primerjalnih grafov za povprečno uspešnost algoritma *GraphGibbs* na skupinah generiranih množic, ki predstavlja povpečje med občutljivostjo in PPV algoritma na vzorčnem nivoju po formuli 4.13.




Slika 5.11: Primerjalni grafi za prikaz korelacijskega koeficienta algoritma *GraphGibbs* na nukleotidnem nivoju med dolžinami motiva $W \in \{6, 9, 13, 18\}$ glede na stopnjo variabilnosti $dv \in \{0, 1, 2, 3\}$, ki so ločeni glede na porazdelitev vzorčnih nizov $P \in \{u, nu\}$.

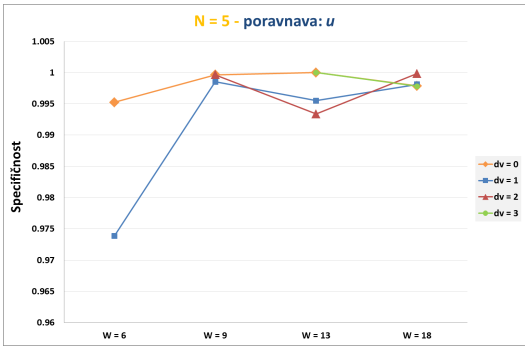
Na sliki 5.12 so prikazani grafi za pregled specifičnosti algoritma *GraphGibbs* na različnih skupinah množic. Generirane množice smo razporedili v šest skupin glede na število sekvenc $N$ v množici in porazdelitvijo pojavitev motiva $P$. Za posamezno skupino so z 3D histogrami podane vrednosti specifičnosti glede na dolžino motiva $W$ in stopnjo variabilnosti $dv$. Specifičnost algoritma meri kako dobro metoda

klasificira dele sekvenc, ki pripadajo ozadju. Drugače povedano, meri kako dobro algoritem določi delež negativnih zadetkov algoritma. Boljša je klasifikacija, večji je delež, torej je vrednost bližje 1. Za izračun specifičnosti uporabimo formulo 4.10.

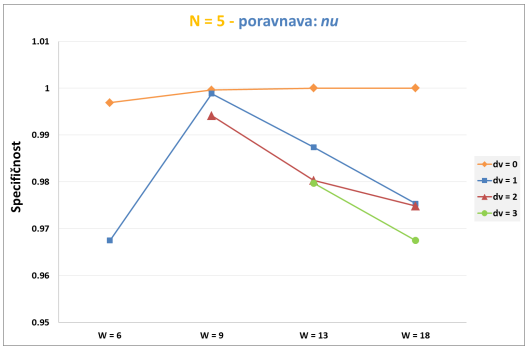
Za stopnjo variabilnosti $dv = 0$ je pri vseh šestih skupinah in vseh dolžinah motiva specifičnost enaka 1 oziroma pri dolžini $W = 6$ skoraj enaka 1. Opazimo lahko tudi precej enakomeren porast vrednosti od skupin $W = 6$ k skupinam $W = 18$ pri skupinah s porazdelitvijo $u$. Kljub (marginalno) različnim skalam lahko opazimo podobno višino istoležnih stolpcev, posebno pri grafih (c) in (e). Pri vseh grafih na levi strani slike 5.12 vidimo, da je največji upad pri množicah z motivom dolžine $W = 6$, kjer se zapis vzorčnih nizov razlikujejo v eni črki.

Večja variabilnost v velikosti specifičnosti se izkaže pri množicah z neenkomerno porazdeljenimi pojavitvami motiva, ki so prikazane na desni strani slike 5.12. Z grafa (b) lahko razberemo, da pri $N = 5$ ima algoritem najboljšo specifičnost pri množicah z dolžino motiva $W = 9$. Pri ostalih dolžinah motivov, ki niso popolnoma ohranjeni, je opazno večji upad.

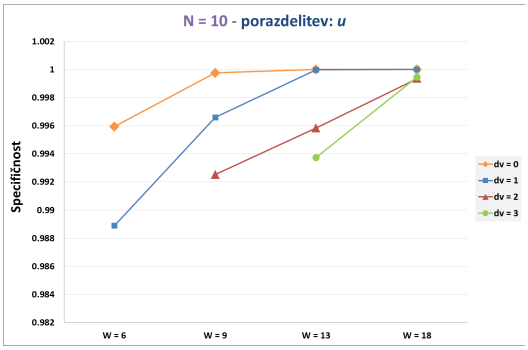
Na podlagi zgornjih rezultatov lahko povzamemo, da se algoritem v povprečju najboljše odrezal pri množicah z motivi dolžine $W = 13$, kjer ni prišlo do velike razpršenosti vrednosti statistik pri večanju stopnje variabilnosti in med obema vrstama porazdelitve. Kadar je porazdeljenost pojavitev motiva enakomerna, se je algoritem najbolje odrezal pri množicah z motivi dolžine $W = 18$. Če primerjamo uspešnost algoritma med množicami z različnim štvilom sekvenc, je najmanjša razpršenost rezultata pri množicah s številom sekvenc $N = 10$ po vseh primerjalnih grafih oziroma glede na razne ločevalne parametre. To nakazuje na boljšo uspešnost algoritma, ko imajo množice do deset danih sekvenc.



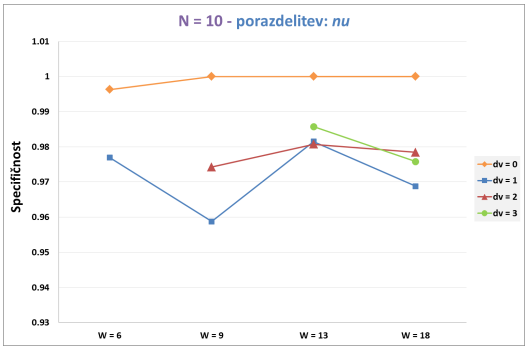

(a) Generirani podatki: $N = 5$ & $P = u$.



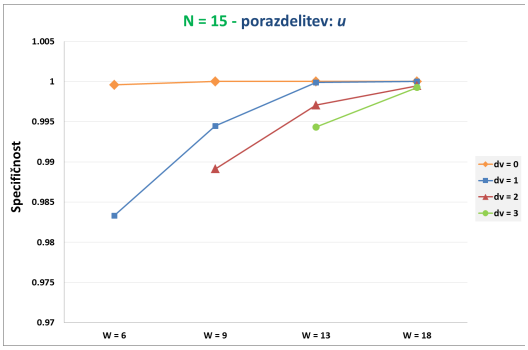

(b) Generirani podatki: $N = 5$ & $P = nu$.



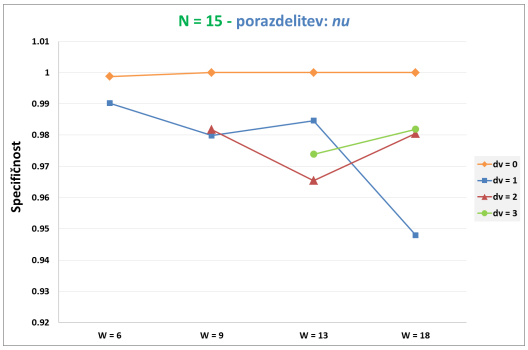

(c) Generirani podatki: $N = 10$ & $P = u$.




(d) Generirani podatki: $N = 10$ & $P = nu$.




(e) Generirani podatki: $N = 15$ & $P = u$.




(f) Generirani podatki: $N = 15$ & $P = nu$.

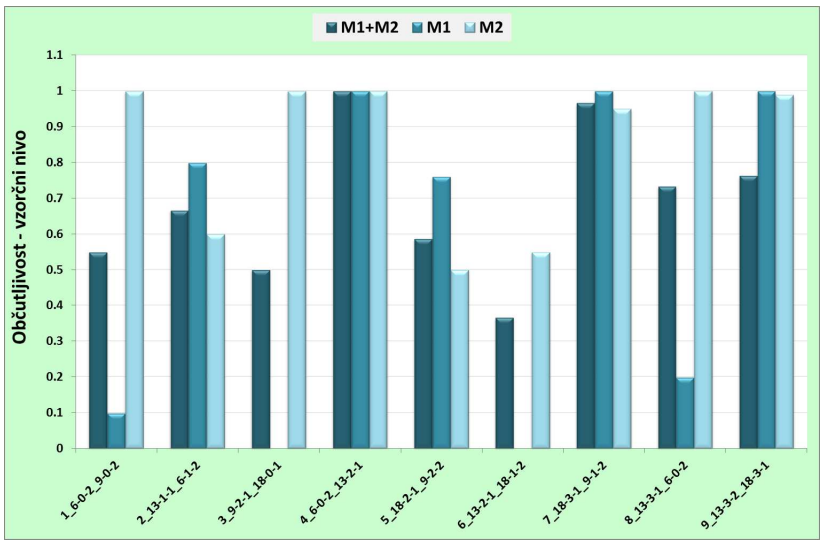
Slika 5.12: Histogrami za prikaz specifičnosti na nukleotidnem nivoju, ki so ločeni po številu sekvenc $N \in \{5, 10, 15\}$ (vrsta) in porazdelitve pojavitev motiva $P \in \{u, nu\}$ (stolpec).

### Množice z dvema motivoma

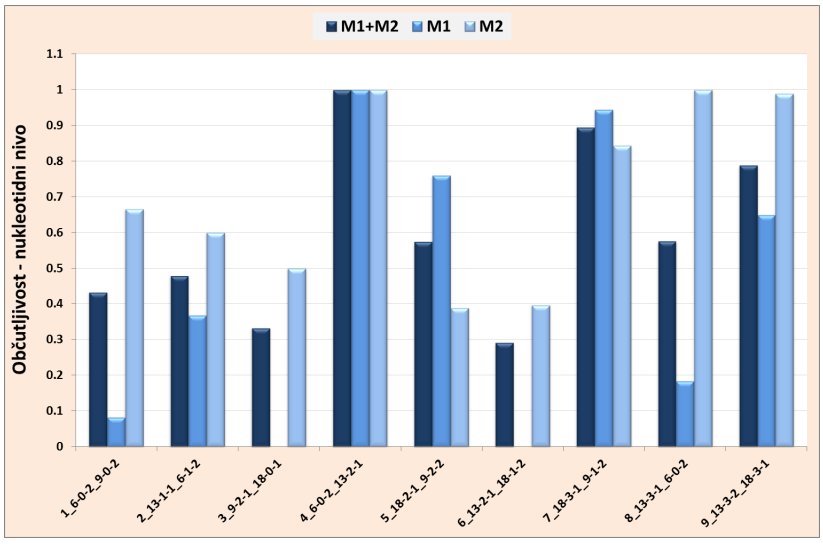
Algoritem *GraphGibbs* smo testirali tudi na generiranih množicah z dvema različnima motivoma. Osnovne lastnosti teh množic so podane v tabeli 4.2. Množice se razlikujejo glede na dolžino in število sekvenc ter po lastnostih motivov. Motiva v posamezni množici se razlikujeta glede na dolžino, stopnjo variabilnosti, števila in porazdelitvijo pojavitev motiva. Množice so označene kot

$$\sharp\_W_1 - dv_1 - E_1\_W_2 - dv_2 - E_2.$$

Število množice $\sharp$ sovpada z zaporednim številom v tabeli 4.2. Trojček števil zazna-




(a) Občutljivost - vzorčni nivo.




(b) Občutljivost - nukleotidni nivo.

Slika 5.13: Občutljivost algoritma *GraphGibbs* na množicah z dvema motivoma: (a) vzorčni nivo in (b) nukleotidni nivo.

muje dolžino-stopnjo variabilnosti-število pričakovanih pojavitev motiva $M_1$ in $M_2$.

Kot pri ostalih generiranih množicah smo tudi na teh množicah algoritem *GraphGibbs* pognali desetkrat in za grafično obdelavo vzeli povprečje desetih rezultatov. Edina prosta parametra sta bila dolžini motivov in število pričakovanih pojavitev posameznega motiva. Ostali parametri algoritma so bili nastavljeni na iste vrednosti kot za množice z enim znanim motivom.

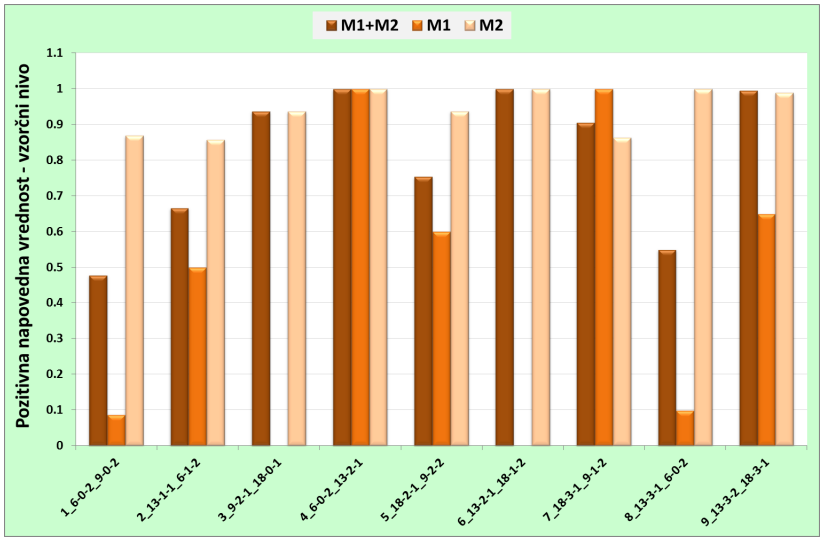
Na sliki 5.13 sta grafa za prikaz občutljivosti algoritma na množice z dvema motivoma, in sicer na grafu (a) je občutljivost na vzorčnem nivoju in na grafu (b) občutljivost na nukleotidnem nivoju. Stopnja občutljivosti je prikazana na ordinatni osi, medtem ko je na abscisi podanih vseh devet množic. K vsaki množici spadajo trije stolpci, kjer prvi stolpec z najtemnejšim odtenkom predstavlja kumulativno mero občutljivosti algoritma na oba motiva hkrati; sosednji stolpec predstavlja občutljivost za prvi motiv ($M_1$); in zadnji stolpec z najsvetlejšim barvnim odtenkom prestavlja občutljivost algoritma na zaznavanje drugega motiva ($M_2$) v dani množici.

Občutljivost meri delež najdenih vzorčnih nizov, ki so tudi v množici znanih vzorčnih nizov. Na vzorčnem nivoju je pri osmih množicah kumulativni delež večji ali enak 0.5, kar pomeni, da je v teh množicah algoritem zaznal več kot polovico znanih pojavitev obeh motivov. Zaznava posameznega motiva je odvisna predvsem od dolžine motiva, stopnje ohranjenosti in zaporedja iskanja posameznega motiva. Iz grafa (b) na sliki 5.13 lahko vidimo, da občutljivost algoritma na nukleotidnem nivoju upade skoraj povsod. Kljub zaznavi večine znanih vzorčnih nizov na vzorčnem nivoju, ti niso bili zaznani v celoti.

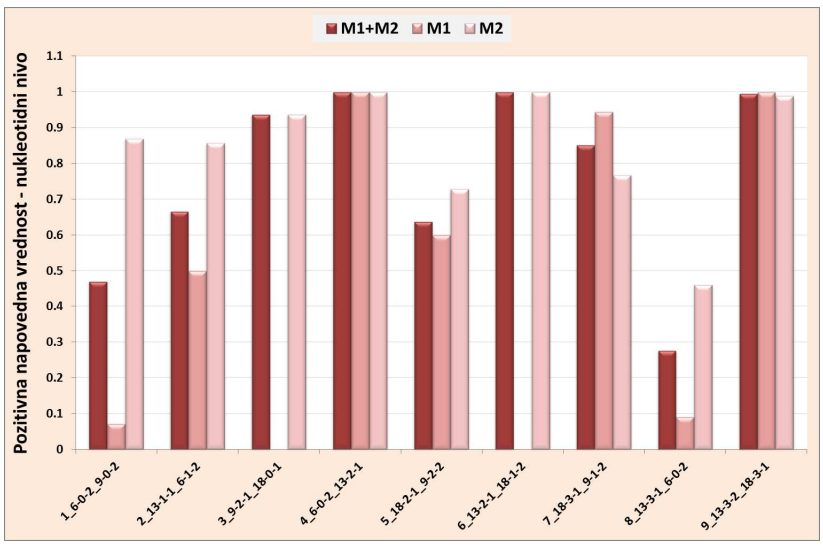
V primeru množice ♯1 z dvema popolnoma ohranjenima motivoma, ki se razlikujeta samo po dolžini, lahko iz obeh grafov razberemo vpliv vrstnega reda na iskanje motivov. Kljub popolni ohranjenosti motivov, je algoritem zaznal vse vzorčne nize samo pri drugem motivu. Eden izmed vzrokov za takšen rezultat je tudi uporabniška nastavitev algoritma *GraphGibbs*. V tem primeru je algoritem najprej zaznal motiv $M_2$, a je upošteval uporabniško nastavitev $W_1 = 6$. Algoritem je zaznal vse znane vzorčne nize motiva $M_2$, kar vidimo na sliki 5.13 ($a$), vendar jih zaradi predvidene krajše dolžine $W_1$ ni zaznal v celoti ($b$). Iskanje motivov v prvem delu algoritma *GraphGibbs* je odvisno od določene dolžine motiva in pričakovanega števila pojavitev. To določa pogoje, pri katerih je najden popolnoma ohranjen vzorec, tudi kandidat za rešitev. Kadar je dejanska dolžina ($W = 6$) krajša kot določena ($W = 9$), potem pogoji niso izpolnjeni in algoritem iskanje konča. Algoritem je $M_2$ v množici ♯1 dejansko zaznal, vendar najdeni vzorec ni izpolnil pogojev (uporabnika), zato je iskanje končal.

To se je ponovilo tudi pri množicah ♯3 in ♯6, kjer velja $W_1 < W_2$ in $E_1 \leq E_2$. Pri množicah ♯4 in ♯9 sicer velja $W_1 < W_2$, vendar je $E_1 > E_2$, kar izboljša izbor pravih vzorčnih množic v prvem delu algoritma *GraphGibbs*. Pri preostalih množicah lahko vidimo še vpliv stopnje variabilnosti posameznega motiva, kajti lahko velja $W_1 > W_2$ in $E_1 < E_2$, a je stopnja variabilnosti $dv_1 \geq dv_2$. Vendar rezultatov na posameznih množic ne moremo direktno primerjati, saj so pomemben dejavnik še preostale lastnosti množic kot sta $N$ in $L$.

Pozitivno napovedno vrednost algoritma *GraphGibbs* na množicah z dvema motivoma smo prikazali na sliki 5.14 na vzorčnem nivoju z grafom (a), in na nukleotidnem nivoju z grafom (b). S to statistiko merimo delež najdenih vzorčnih nizov, ki so znani. Če na vzorčnem nivoju primerjamo grafa (a) na slikah 5.13 in 5.14, torej občutljivost in PPV na vzorčnem nivoju, lahko sklepamo, da algoritem poda manj lažno pozitivnih zadetkov kot lažno negativnih zadetkov na množicah z dvema motivoma. To pomeni, da algoritem ne oblikuje prevelikih vzorčnih množic, ki so preveč heterogene oziroma s preveliko stopnjo variabilnosti. Najdeni vzorčni nizi so torej večinoma znani.




(a) Pozitivna napovedna vrednost - vzorčni nivo.



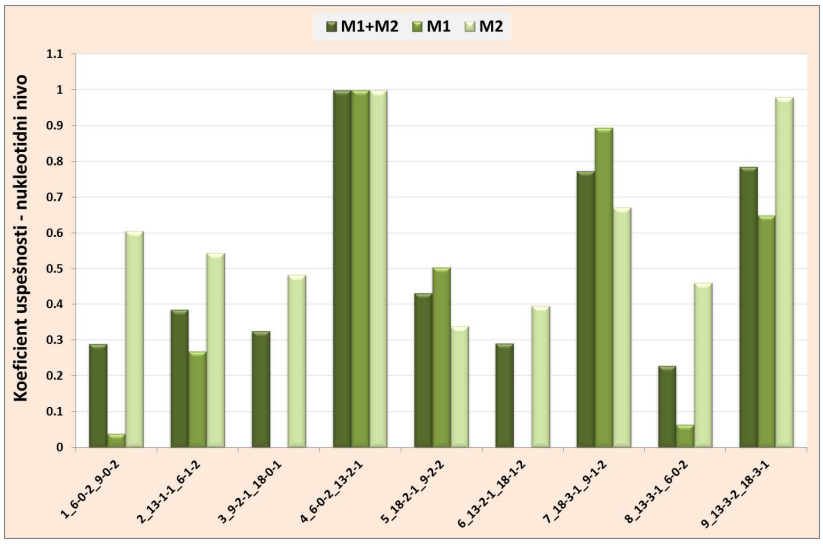

(b) Pozitivna napovedna vrednost - nukleotidni nivo.

Slika 5.14: Pozitivna napovedna vrednost algoritma *GraphGibbs* na množice z dvema motivoma: (a) vzorčni nivo in (b) nukleotidni nivo.

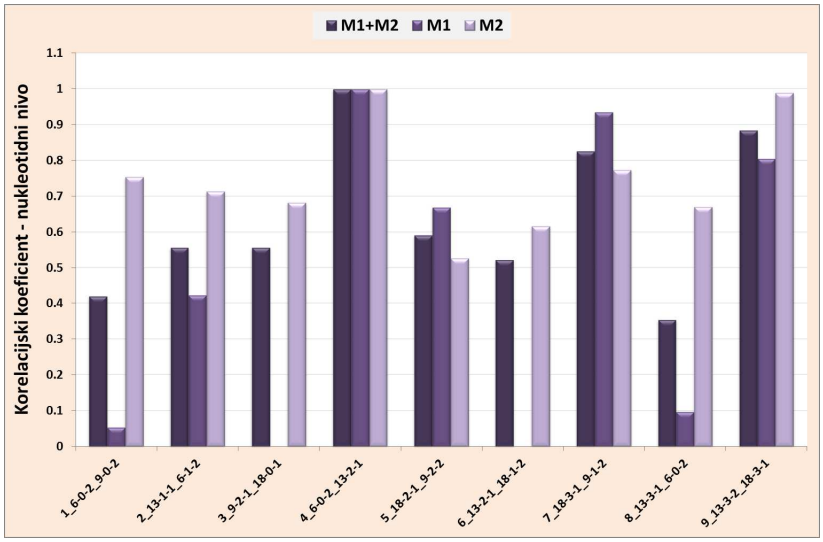
Na vzorčnem nivoju je pozitivna napovedna vrednost za oba motiva hkrati višja ozi-

roma enaka 0.5 pri vseh množicah razen množice ♯1. Pri množicah ♯2, ♯4, ♯5, ♯7 in ♯9 je PPV visok tudi za vsak motiv posebej. Pri osmi množici ima prvi motiv visoko stopnjo variabilnosti, medtem ko je drugi motiv popolnoma ohranjen. Pri tretji in šesti množici pa drugi motiv sploh ni bil podan kot rešitev. Povprečna uspešnost algoritma *GraphGibbs* na vzorčni ravni pa je prikazana na sliki C.8 v prilogi C.2.

Na nukleotidnem nivoju, ki ga prikazuje graf (b) na sliki 5.14, je PPV marginalno nižja od vrednosti na vzorčni ravni. Glede na grafa (b) na slikah 5.13 in 5.14 lahko povzamemo, da je število znanih vzorčnih nizov, ki so bili najdeni, v povprečju enako polovici znane vzorčne množice, vendar so bili le ti zaznani v večini, ali drugače presek najdenega znanega in znanega vzorčnega niza je velik.




(a) Koeficient uspešnosti - nukleotidni nivo.

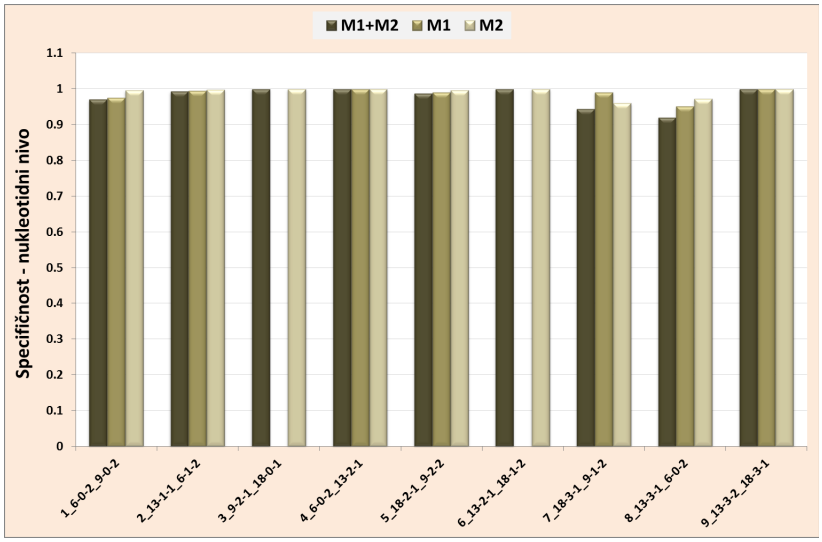



(b) Korelacijski koeficient - nukleotidni nivo.

Slika 5.15: Množice z dvema motivoma na nukleotidne nivoju: (a) koeficient uspešnosti in (b) korelacijski koeficient.

Na nukleotidnem nivoju si lahko ogledamo še koeficient uspešnosti in korelacijski koeficient. Prvi zrcali naša opažanja stopnje občutljivosti algoritma in njegove pozitivne napovedne vrednosti na vzorčnem nivoju, medtem ko korelacijski koeficient nakazuje pozitivno korelacijo med znano in najdeno poravnavo. Koeficient uspešnosti je nižji predvsem zaradi lažno negativnih zadetkov, torej tistih pojavitev motivov, ki jih algoritem ni zaznal, kar se je odražalo tudi pri občutljivosti algoritma. Vendar je med najdenimi in znanimi vzorčnimi nizi relativno visoka pozitivna povezanost, kar nakazuje, da so pravilno določeni vzorčni nizi zaznani v večjem delu na nukleotidnem nivoju.

Poleg koeficienta uspešnosti in korelacije lahko na nukleotidnem nivoju izračunamo še specifičnost algoritma na množicah z dvema motivoma. Z grafa na sliki 5.16 lahko razberemo, da algoritem dobro določa ozadje sekvenc kot ozadje pri večini množic. To nam pove, da algoritem *GraphGibbs* redko določi veliko število lažno pozitivnih vzorčnih nizov glede na dolžino in število vseh sekvenc v množici.




Slika 5.16: Prikaz specifičnosti algoritma *GraphGibbs* na množicah z dvema motivoma.

### Analiza konvergence in odstotkov ujemanja

Pri testiranju smo na vsaki množici pognali algoritem *GraphGibbs* desetkrat, da smo lahko preverili ponovljivost uspešnosti algoritma. Parameter $M$ je nedoločen, zato lahko algoritem najde več različnih motivov v dani množici pri posameznem poskusu. Rešitev enega poskusa je motiv, katerega poravnava se prekriva s poravnavo znanega motiva in jo ocenimo z različnimi statistikami ter odstotkom ujemanja presečnega vzorca rešitve in znanega motiva. Rešitev desetih ponovitev poskusa ocenimo s povprečnimi vrednostmi posameznih zadetkov $TP$, $TN$, $FP$ in $FN$, medtem ko za njen odstotek ujemanja vzamemo povprečje odstotkov ujemanja rešitve posamezne ponovitve.

V tabeli 5.2 predstavimo uspešnost algoritma na generiranih množicah glede na odstotek ujemanja $U$. Pri večini množic ($\approx 94\%$) je algoritem zaznal znan motiv v celoti kot edino rešitev ali kot eno izmed možnih rešitev. Pri nekaj generiranih množicah ($\approx 3\%$) algoritem ni zaznal nobenega znanega vzorčnega niza. Pri ostalih množicah ($\approx 3\%$) pa je algoritem zaznal znan motiv pri več različnih rešitvah. To pomeni, da je del znanih vzorčnih nizov algoritem klasificiral pod eno vzorčno množico oziroma pri eni rešitvi, ostanek pa pri drugi rešitvi v isti ponovitvi poskusa. Pri vsaki od možnosti smo podali še meji $\geq 50\%$ in $< 50\%$, s katerima smo definirali dve skupini glede na velikost ujemanja najdenega in znanega motiva. Pri večini generiranih množic je algoritem podal eno rešitev, zato smo te množice podrobneje pogledali, in sicer smo preverili, kakšen je delež množic, kjer je ujemanje znanega in najdenega motiva popolno (100%). V zadnji vrstici so kumulativni deleži vseh generiranih množic.

Za večino množic ($\approx 67\%$), kjer je algoritem zaznal znan motiv v eni izmed najdenih rešitev, je bilo ujemanje popolno. Torej znan in najden motiv (presečna vzorca) se popolnoma prekrivata. To velja za množice z daljšim motivom, to je pri $W = 13$ in $W = 18$. Prav tako lahko opazimo, da je popolno ujemanje pri vseh množicah s popolno ohranjenimi motivi, razen pri množicah z $N = 5$. Pri slednjih je pri kakšni množici algoritem podal rešitev z nepopolnim ujemanjem, kar je posledica prvega dela algoritma, ko algoritem gradi motiv.

Pri množicah s krajšimi motivi so množice bolj porazdeljene med posamezne skupine glede na odstotek ujemanja. Razpršenost je posledica višje stopnje variabilnosti $dv \neq 0$, kjer so lahko poravnave najdenih rešitev zamaknjene glede na poravnavo znanega motiva. Učinek višje stopnje variabilnosti med motivi reguliramo z Gibbsovim vzorčevalnikom, kjer na podlagi popolno ohranjenih nizov iz prvega dela, algoritem konvergira k poravnavi, ki ima najvišjo vrednost statistike $I$. Večja je variabilnosti med znanimi vzorčnimi nizi, krajši bo popolnoma ohranjen podniz. Zelo kratki nizi dolžine $3-6$ baznih parov pa imajo lahko slučajne pojavitve v dani množici, ki pa ne sovpadajo s poravnavo znanega motiva in tako preusmerijo iskanje Gibbsovega vzorčevalnika na drug lokalni del prostora rešitev.

Pri nekaterih množicah algoritem *GraphGibbs* ni zaznal nobenega od znanih vzorčnih nizov. Kljub temu smo preverili, kakšen je odstotek ujemanja $U$. Pri realnih podatkih nimamo nujno vseh vzorčnih nizov določenih, ki so dejansko del istega motiva, in jih bo zato algoritem klasificiral kot negativne zadetke. Visoko ujemanje najdenega motiva z znanim pa lahko nakazuje na obstoj vzorčnih nizov, ki so bili spregledani pri analiziranju sekvenc. Takšne indukcije delamo lahko na realnih podatkih in ne na generiranih, kjer se generacija sekvenc in vzorčnih nizov izvaja v kontroliranem okolju.

Pri generiranih testnih množicah smo tako dodatno naredili parni $t$-test in preverili ponovljivost rezultatov na slučajno izbranih množicah. Naredili smo štiri glavne skupine glede na dolžino motiva $W$ in dodatno ločili množice glede na stopnjo variabilnosti $dv$. Pri vsaki podskupini smo glede na število sekvenc $N$ v množici slučajno izbrali tri množice, ki se razlikujejo glede na porazdelitev pojavitev motiva, števila

Tabela 5.2: Delež množic glede na odstotek ujemanja $U$. Deleži in meje so izraženi v odstotkih.

| | | | ena rešitev (%) | | | več rešitev (%) | | ni rešitve (%) | |
|---|---|---|---|---|---|---|---|---|---|
| ***W*** | ***dv*** | ***N*** | **100** | **≥ 50** | **< 50** | **≥ 50** | **< 50** | **≥ 50** | **< 50** |
| **6** | 0 | 5 | **85.71** | | | 7.14 | 7.14 | | |
| | | 10 | **83.33** | | | 16.67 | | | |
| | | 15 | **100** | | | | | | |
| | 1 | 5 | | 33.33 | 13.33 | 33.33 | 20 | | |
| | | 10 | 36.36 | 36.36 | | 18.18 | | 9.09 | |
| | | 15 | 16.67 | **66.67** | | 8.33 | 8.33 | | |
| **9** | 0 | 5 | **87.50** | 12.5 | | | | | |
| | | 10 | **100** | | | | | | |
| | | 15 | **100** | | | | | | |
| | 1 | 5 | **66.67** | 25 | | | | | 8.33 |
| | | 10 | 41.67 | **58.33** | | | | | |
| | | 15 | 41.67 | **58.33** | | | | | |
| | 2 | 5 | **56.25** | | 18.75 | | | | 25 |
| | | 10 | 31.25 | **50** | 6.25 | | | | 12.50 |
| | | 15 | **62.5** | 37.5 | | | | | |
| **13** | 0 | 5 | **100** | | | | | | |
| | | 10 | **100** | | | | | | |
| | | 15 | **100** | | | | | | |
| | 1 | 5 | **50** | 18.75 | 18.75 | | | | 12.5 |
| | | 10 | **75** | 6.25 | 18.75 | | | | |
| | | 15 | **87.5** | | 12.5 | | | | |
| | 2 | 5 | **50** | 12.5 | 18.75 | | | | 18.75 |
| | | 10 | **56.25** | 18.75 | 18.75 | | | | 6.25 |
| | | 15 | **62.5** | 18.75 | 18.75 | | | | |
| | 3 | 5 | **50** | 6.25 | 37.5 | | | | 6.25 |
| | | 10 | **81.25** | 12.5 | 6.25 | | | | |
| | | 15 | **60** | 26.67 | 13.33 | | | | |
| **18** | 0 | 5 | **87.5** | | 6.25 | | | | 6.25 |
| | | 10 | **100** | | | | | | |
| | | 15 | **100** | | | | | | |
| | 1 | 5 | **50** | 18.75 | 31.25 | | | | |
| | | 10 | **75** | | 25 | | | | |
| | | 15 | **60** | | 40 | | | | |
| | 2 | 5 | **56.25** | 12.5 | 18.75 | | | | 12.5 |
| | | 10 | **56.25** | 31.25 | 12.50 | | | | |
| | | 15 | **60** | 33.33 | 6.67 | | | | |
| | 3 | 5 | 37.5 | 37.5 | 25 | | | | |
| | | 10 | **56.25** | 18.75 | 25 | | | | |
| | | 15 | **66.67** | 20 | 13.33 | | | | |
| **Skupaj** | | | **66.95** | **16.44** | **10.68** | **1.86** | **0.85** | **0.17** | **3.05** |

pričakovanih pojavitev ter po dolžini sekvenc. Tako smo v vsaki podskupini zbrali devet množic. Rezultati $t$-testa in testa ponovljivosti na trinajstih skupinah so podani v tabeli 5.3.

S parnim $t$-testom smo primerjali vrednosti statistike $I$ znane poravnave in najdene poravnave na isti množici, da bi ugotovili statistično pomembne razlike med skupinama. Statistika $I$ najdene poravnave je mediana vrednosti statistike $I$ desetih rešitev na isti množici. Edina večja statistično pomembna razlika se je izkala pri skupini $W = 13$ in $dv = 3$. Pri dveh primerih, in sicer $W = 6$, $dv = 0$ ter $W = 18$, $dv = 2$ je $p$-vrednost na meji. Vse navedene množice so v tabeli 5.3 odebeljene. V primeru, ko so bile vrednosti statistike $I$ pri znani in najdeni poravnavi identični, statistične razlike nismo mogli določiti, kar smo v tabeli 5.3 označili z *nan*.

Tabela 5.3: Rezultati parnega $t$-testa in preverjanja ponovljivosti na slučajno izbranem vzorcu po skupinah množic, ki jih ločimo glede na $W$ in na pripadajočo stopnjo variabilnosti $dv$.

| **Skupina** | ***dv*** | ***t*-test** | **% RSD** |
|---|---|---|---|
| **6** | 0 | **0.0468** | 3.546829 |
| | 1 | 0.3418 | 1.200425 |
| **9** | 0 | 0.3466 | 0.90211 |
| | 1 | 0.1284 | 7.354902 |
| | 2 | 0.1863 | 2.151753 |
| **13** | 0 | nan | 0 |
| | 1 | 0.1204 | 0.340921 |
| | 2 | 0.0527 | 5.81568 |
| | 3 | **0.0131** | 5.003612 |
| **18** | 0 | 0.3466 | 0 |
| | 1 | 0.0892 | 0.191793 |
| | 2 | **0.0447** | 2.314084 |
| | 3 | 0.2016 | 1.305416 |

Ponovljivost metode opiše kakovost ujemanja rezultatov pri istih pogojih, torej isto metodo, isti testni material in isti merski inštrument [55]. Pri naši raziskavi smo testirali vsako množico podatkov desetkrat z istim algoritmom (*GraphGibbs*) na istem računalniku. Ponovljivost izrazi spodnjo mejo variabilnosti natačnosti (ang. precision) dobljenih rezultatov. Rezultate ponovljivosti smo izrazili v obliki relativnega standardnega odklona (ang. Relative Standard Deviation). Manjši je relativni odstotek standardnega odklona, bolj so ponovljivi rezultati algoritma, ali drugače: algoritem na isti množici najde isto rešitev oziroma poravnavo, ki je blizu pravi poravnavi motiva.

Z izjemo skupine $W = 6$ je $\%RSD$ pri ostalih skupinah na množicah s popolnoma ohranjenimi motivi pod 1%. Pri najdajših dveh dolžinah pa celo 0. V teh

primerih je natančnost rezultatov posamezne ponovitve pričakovano dobra. Pri skupini $W = 6$ je bil največji odstotek $\%RSD$ ravno pri $dv = 0$, kjer smo s $t$-testom določili marginalno statistično razliko med statistikama $I$. To je posledica delovanja prvega dela algoritma *GraphGibbs*. Namreč pri večini primerov, kar je razvidno iz primerjalnih grafov za PPV algoritma 5.8 kot tudi iz krožnih grafov na sliki C.12, je pri množicah z $dv = 0$ algoritem podal rešitev dobljeno v prvem delu, kjer pa je lahko bilo določenih več pojavitev motiva kot jih je dejansko. Število vzorčnih nizov pa vpliva na velikost statistike $I$. Rezultat $t$-testa in vrednosti $\%RSD$ temeljita na razlikah med statistikami $I$.

Pri množicah z motivi dolžine $W = 9$ je najvišja vrednost $\%RSD$ pri podskupini množic, kjer je $dv = 1$. Skladno s tem je tudi $p$-vrednost $t$-testa pri tej podskupini najnižja. Približno sedem odstotkov je tudi najvišja vrednost $\%RSD$ med vsemi danimi skupinami množic. Pri daljših motivih je navečji odstotek $\%RSD$ pri podskupinah množic s stopnjo variabilnosti $dv = 2$. Takšne lastnosti podatkovnih množic so se za iskanje motivov z algoritmom *GraphGibbs* s stališča ponovljivosti pokazale kot najmanj primerne.

Sedaj si pa oglejmo še razporeditev generiranih množic glede na način določitve motiva, ki jo predstavimo s tabelo 5.4. Znan motiv lahko algoritem zazna in najde pravo poravnavo na več načinov, in sicer s prvim delom ali pa z Gibbsovim vzorčevalnikom. Rešitvi posameznega dela sta lahko usklajeni popolnoma, deloma ali pa je Gibbsov vzorčevalnik skonvergiral k drugemu lokalnemu ekstremu od začetne točke, ki smo jo določili s prvim delom. Recimo, da algoritem v prvem delu določi $m$ različnih motivov, potem se lahko v skrajnem primeru dogodi, da Gibbsov vzorčevalnik za vsako začetno točko konvergira k novemu lokalnemu ekstremu. Tako dobimo tudi do $2m$ rešitev pri enem pogonu algoritma. Takšni primeri so redki, saj bi to pomenilo, da je množica preveč heterogena.

V večini primerov algoritem *GraphGibbs* določi število motivov že v prvem delu (Graph) in nekaj dodatnih lokalnih ekstremov z Gibbsovim vzorčevalnikom (Gibbs). V tabeli 5.4 je delež vseh obdelanih generiranih množic glede na opisane različne določitve znanega motiva. Posebej smo obravnavali množice, kjer so znani vzorčni nizi bili razdeljeni med dvema ali več rešitev v posamezni ponovitvi poskusa, in množice, kjer ni bilo znane rešitve.

Kadar pri prvem delu algoritem zazna popolnoma ohranjen niz v dani množici glede na uporabnikove vrednosti za dolžino motiva in število pojavitev, potem se Gibbsov vzorčevalnik niti ne zažene. Takšen dogodek smo označili kot Graph-nan, kjer kratica nan predstavlja nedefinirano poravnavo Gibbsovega vzorčevalnika. Iz tabele 5.4 lahko razberemo, da je pri večini množic s popolnoma ohranjenim motivom zadostoval prvi del algoritma za določitev rešitve.

Pri množicah z motivi s pozitivno stopnjo variabilnosti je algoritem dosegel rešitev s pomočjo Gibbsovega vzorčevalnika. V nekaj primerih, predvsem pri množicah z dolžinama motivov $W = 6$ in $W = 9$, pa je rešitev prvega dela (Graph) boljša kot rešitev, ki smo jo dobili v drugem delu (Gibbs). Torej je zaznava motiva na

Tabela 5.4: Deleži množic glede na način določitve motiva izraženi v odstotkih.

| *W* | *dv* | *N* | Graph-nan | Graph | Gibbs | Več rešitev | Ni rešitve |
|---|---|---|---|---|---|---|---|
| **6** | 0 | 5 | **78.57** | 7.14 | | 14.29 | |
| | | 10 | **75** | 8.33 | 8.33 | 8.33 | |
| | | 15 | **100** | | | | |
| | 1 | 5 | 6.67 | 13.33 | 20 | **60** | |
| | | 10 | | **54.55** | 9.09 | 27.27 | 9.09 |
| | | 15 | | **83.33** | | 16.67 | |
| **9** | 0 | 5 | **81.25** | 12.5 | 6.25 | | |
| | | 10 | **81.25** | | 18.75 | | |
| | | 15 | **100** | | | | |
| | 1 | 5 | 8.33 | 33.33 | **50** | | 8.33 |
| | | 10 | | 41.67 | **58.33** | | |
| | | 15 | | **58.33** | 41.67 | | |
| | 2 | 5 | 12.5 | 37.5 | 25 | | 25 |
| | | 10 | | **50** | 31.25 | | 18.75 |
| | | 15 | 6.25 | **62.5** | 31.25 | | |
| **13** | 0 | 5 | **56.25** | 25 | 18.75 | | |
| | | 10 | **81.25** | 12.5 | 6.25 | | |
| | | 15 | **93.75** | | 6.25 | | |
| | 1 | 5 | 18.75 | | **68.75** | | 12.5 |
| | | 10 | 12.5 | 12.5 | **75** | | |
| | | 15 | 12.5 | | **87.5** | | |
| | 2 | 5 | 12.5 | 18.75 | **50** | | 18.75 |
| | | 10 | | 31.25 | **62.5** | | 6.25 |
| | | 15 | 6.25 | 12.5 | **81.25** | | |
| | 3 | 5 | 12.5 | 18.75 | **62.5** | | 6.25 |
| | | 10 | | 31.25 | **62.5** | | 6.25 |
| | | 15 | | 33.33 | **66.67** | | |
| **18** | 0 | 5 | **62.5** | 25 | 6.25 | | 6.25 |
| | | 10 | **87.5** | | 12.5 | | |
| | | 15 | **93.75** | 6.25 | | | |
| | 1 | 5 | 31.25 | | **68.75** | | |
| | | 10 | 18.75 | 6.25 | **75** | | |
| | | 15 | 43.75 | | **56.25** | | |
| | 2 | 5 | 6.25 | | **81.25** | | 12.50 |
| | | 10 | 12.5 | | **87.5** | | |
| | | 15 | 6.67 | | **93.33** | | |
| | 3 | 5 | 12.5 | 6.25 | **81.25** | | |
| | | 10 | 18.75 | 6.25 | **75** | | |
| | | 15 | 13.33 | | **86.67** | | |
| **Skupaj** | | | **32.88** | **17.12** | **44.07** | **2.71** | **3.22** |

vzorčnem in nukleotidnem nivoju s poravnavo rešitve prvega dela uspešnejša kot s poravnavo rešitve drugega dela. Kjer so bili motivi daljši in relativno ohranjeni, pa ja algoritem najboljšo rešitev dosegel s pomočjo Gibbsovega vzorčevalnika.

Pri dolžini relativno ohranjenega motiva $W = 6$ in množicah s petimi sekvencami pa je algoritem pri večjem deležu množic (60%) zaznal znani motiv v več najdenih rešitvah. Motivi teh množic so kratki in imajo še krajši popolnoma ohranjen podniz, ki je presek vseh znanih vzorčnih nizov. Zelo kratki nizi se večkrat pojavijo v dani množici, saj se kratek niz pojavi tako pri znanih vzorčnih nizih kot tudi slučajno znotraj množice. To pa lahko vpliva na gradnjo motiva v prvem delu algoritma, posebno kadar je število znanih vzorčnih nizov majhno, kot je v primeru množic s petimi sekvencami, kjer je bilo največ deset znanih vzorčnih nizov.

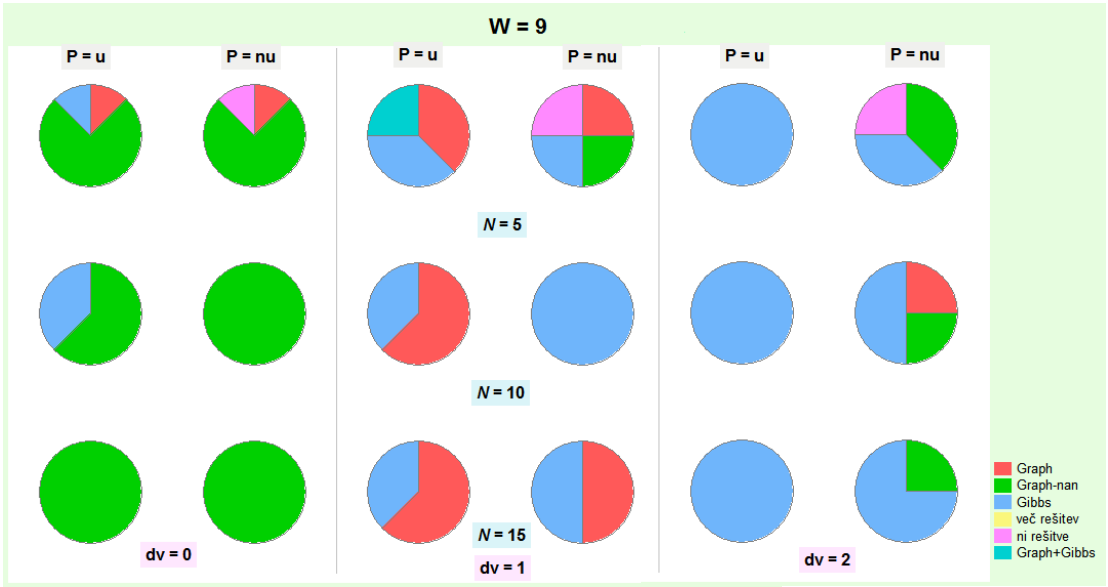
Pri večini generiranih množic je torej algoritem *GraphGibbs* našel optimalno rešitev z uporabo prvega in drugega dela algoritma. S prvim delom določimo en del znane vzorčne množice, ki jo dopolnimo s pomočjo drugega dela. Prvi del tako zagotavlja zaznavo vsaj enega znanega vzorčnega niza, kajti z izjemo nekaj množic s krajšimi motivi, kjer ni bilo znane rešitve, je algoritem našel vsaj eno pojavitev znanega motiva v danih množicah.

Bolj razširjen pregled razporeditve deležov množic glede na način določitve smo predstavili še s krožnim grafikonom. Na sliki 5.17 so prikazani krožni grafikoni za prikaz deležov množic z dolžino motiva $W = 9$. Dodali smo še eno kategorijo, in sicer Graph+Gibbs, ki beleži delež množic, kjer je v desetih ponovitvah algoritem nekajkrat bolje zaznal znano vzorčno množico z rešitvijo prvega dela in nekajkrat z rešitvijo drugega dela. Na takšno delitev v desetih ponovitvah vplivata odgovarjajoča slučajna elementa v obeh delih algoritma.

Vsaka od kategorij je tudi barvno označena za boljši pregled kako so množice porazdeljene po danih kategorijah. Barvna shema je naslednja:

- **Graph** - algoritem bolje zaznal znano vzorčno množico s prvim delom,
- **Graph-nan** - algoritem je našel znano vzorčno množico v prvem delu, a ni zagnal Gibbsovega vzorčevalnika,
- **Gibbs** - algoritem je bolje zaznal znano vzorčno množico z drugim delom,
- **več rešitev** - algoritem je zaznal vzorčne nize pri več različnih rešitvah pri eni ponovitvi,
- **ni rešitve** - algoritem ni zaznal nobene pojavitve znanega motiva in
- **Graph+Gibbs** - algoritem je bolj uspešno zaznal znano vzorčno množico pri nekaterih ponovitvah s prvim delom in pri ostalih z drugim delom.

Pri množicah s popolnoma ohranjenimi motivi ni bilo potrebno pognati Gibbsovega vzorčevalnika, da je algoritem določil znano množico, tako pri enakomerno porazdeljenih pojavitvah motiva kot tudi pri neenakomerni porazdelitvi. V nekaj primerih




Slika 5.17: Krožni diagram delitve množic z motivom dolžine $W = 9$ glede na način določitve rešitve algoritma *GraphGibbs*. V vrstah so prikazani diagrami glede na število sekvenc $N \in \{5, 10, 15\}$ v množicah. Po stopcih so pa v paru po porazdelitvi $P \in \{u, nu\}$ prikazani diagrami glede na stopnjo variabilnosti $dv \in \{0, 1, 2\}$.

pa je algoritem pravo rešitev določil z uporabo vzorčevalnika. Od teh množic se je pri dveh dogodilo, da vzorčevalnik ni skonvergiral k pravi poravnavi, kar pomeni, da je rešitev prvega dela (Graph) bolje zastopala pravo porazdelitev. Pri eni množici z neenakomerno porazdelitvijo znanih vzorčnih nizov pa algoritem ni našel nobene pojavitve motiva.

Pri višjih stopnjah variabilnosti je algoritem pri večini množic bolje zaznal pojavitve znanega motiva tako s prvim delom kot tudi z Gibbsovim vzorčenjem. V nekaj primerih algoritem ni določil nobenega od znanih motivov. Prav tako je pri par množicah algoritem pri različnih ponovitvah testiranja določil rešitev enkrat s prvim in enkrat z drugim delom (Graph+Gibbs). Kljub relativni ohranjenosti motivov pa je algoritem pri manjšem odstotku množic določil rešitev samo s prvim delom (Graph-nan), torej ni niti zagnal Gibbsovega vzorčevalnika. To je posledica več dodatnih, slučajnih pojavitev krajšega popolnoma ohranjenega niza, ki je element preseka vseh znanih vzorčnih nizov.

Krožni diagrami množic z ostalimi dolžinami motiva so podani v prilogi C.4. Na slikah C.15 in C.16 so prikazani krožni diagrami za množice z motivi dolžine $W = 18$. Pretežno so bile množice določene samo s prvim delom (Graph-nan) pri nižjih stopnjah variabilnosti $dv \in \{0, 1\}$ in z drugim delom pri višjih stopnjah variabilnosti $dv \in \{2, 3\}$. Večjo zastopanost Graph-nan načina lahko pripišemo k nastavitvi algoritma *GraphGibbs* v prvem delu, kjer po določitvi najdaljšega možnega motiva algoritem naredi primerjavo pojavativev njegovih krajših podnizov. Kadar je motiv bolj ohranjen, torej ima nižjo stopnjo variabilnosti, lahko število pojavitev krajšega podniza preseže pričakovano število pojavitev $E$, kar pomeni, da je glede na dane

parametre algoritem že našel rešitev in zato tako ni potrebe, da požene Gibbsov vzorčevalnik.

Pri dolžini $W = 13$ je algoritem pri večini primerov razen pri množicah s popolnoma ohranjenimi motivi pognal še Gibbsov vzorčevalnik. Kljub temu pa lahko je s slik C.13 in C.14 razvidno, da je večkrat algoritem bolje zaznal znane vzorčne množice z rešitvijo prvega dela, in sicer pretežno pri neenakomerno porazdeljenih pojavitvah motiva. Deloma lahko takšen rezultat pripišemo k predpostavki, da je v vsaki sekvenci vsaj ena pojavitev motiva, kar vpliva na gradnjo Markovske verige pri Gibbsovem vzorčenju. Posledično je algoritem bolj uspešen v prvem delu, saj je v drugem delu našel drug lokalni ekstrem, pri katerem pa je manjša zaznava znane množice na vzorčnem ali pa na nukleotidnem nivoju.

Največja variabilnosti pri načinu zaznave prave poravnave je pri najkrajši dolžini motiva, to je $W = 6$. Na sliki C.12 vidimo, da je pri popolnoma ohranjenih motivih algoritem potreboval samo prvi del za določitev znane vzorčne množice. Pri nekaterih množicah pa je algoritem zaznal pojavitve motiva pri različnih rešitvah iste ponovitve. Ta način je pogost tudi pri množicah s stopnjo variabilnosti $dv = 1$, predvsem pri neenakomerni porazdelitvi pojavitev motiva. Pri velikem deležu teh množic je prevladala rešitev prvega dela (Graph), predvsem pri množicah z večjim številom sekvenc, kjer je $N = 10$ ali $N = 15$. Drugače je algoritem določil rešitev s pomočjo vzorčevalnika, z izjemo dveh množic, pri katerih je v enem primeru določil rešitev ali v prvem ali v drugem delu (Graph+Gibbs), v drugem primeru pa znane vzorčne množice ni zaznal.

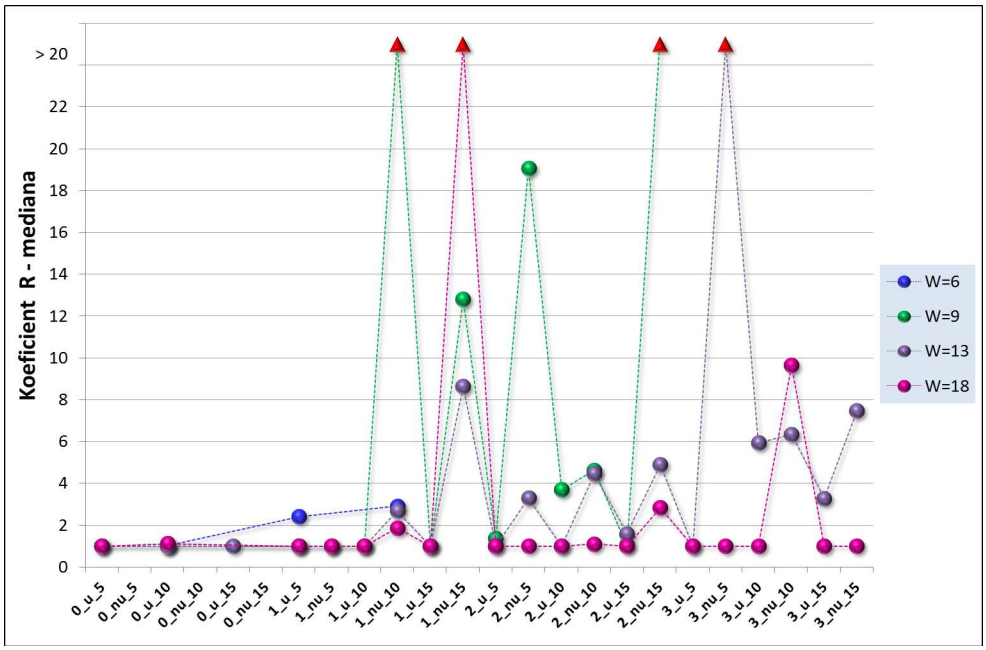
Pri primerih, kjer je algoritem *GraphGibbs* pognal Gibbsov vzorčevalnik lahko naredimo tudi Gelman-Rubinov diagnostični test konvergence vzorčevalnika, ki je opisan v algoritmu 8. Pri vsaki od desetih ponovitvah smo pri testiranju pognali vzorčevalnik desetkrat, da smo dobili deset Markovskih verig pri 250-ih iteracijah. Če upoštevamo polovico iteracij, to je $D = 125$ iteracij, kot obdobje zažiganja, smo dobili deset verig $i = 1, \ldots, 10$ dolžine $T = 125$ za preverjanje stacionarnosti. Poračunali smo povprečje verige $\overline{H}_{i*}$, celotno povprečje $\overline{H}$, varianco med verigami $Var_B$, varianco znotraj posamezne verige $Var_W$ in aposteriorno varianco vseh desetih verig $\hat{\sigma^2}$. Nato smo definirali koeficient $R_{125} = \frac{\hat{\sigma^2}}{Var_W}$. V primeru, ko je vzorčevalnik dosegel stacionarno verigo, je koeficient $R_{125} \approx 1$.

Na sliki 5.18 so prikazani rezultati Gelman-Rubinovega testa na generiranih množicah. Za predstavitev smo množice ločili v štiri skupine glede na dolžino motiva. Na abscisi so predstavljene večje podskupine znotraj teh množic, ki smo jih označili kot

$$dv\_P\_N.$$

V podskupini se tako množice ločije še na pričakovano število pojavitev motiva $E$ in dolžino ozadja sekvenc $L$.

S točkami označimo kumulativno vrednost koeficienta $R$ posamezne skupine, ki smo ga določili kot mediano vrednosti koeficientov $R$ pri posameznih množicah v skupini. Vse vrednosti, ki so večje od 20, nismo eksplicitno prikazali, ampak smo jih




Slika 5.18: Gelman-Rubin test za diagnozo stacionarnosti Markovske verige, ki jo gradi Gibbsov vzorčevalnik pri testiranju generiranih množic.

zaznamovali s trikotnikom. Pri množicah s tako visoko vrednostjo koeficienta $R$ nismo mogli potrditi stacionarnost oziroma konvergenco Gibbsovega vzorčevalnika. Slednje se je izkazalo samo pri štirih skupinah, kjer je pri nekaterih množicah prišlo ali do prevelike razpršenosti znotraj Markovskih verig (divergenca) ali pa med Markovskimi verigami (različni lokalni ekstremi).

Pri ostalih podskupinah pa je koeficient $R$ manjši in pri večini teh tudi blizu vrednosti 1. Dobre rezultate je Gibbsov vzorčevalnik dosegel pri skupini $W = 18$, kjer je z nekaj izjemami, pri večini vrednost mediane koeficintov $R$ posameznih množic enaka oziroma blizu 1, tudi pri višjih stopnjah variabilnosti. Malo večja je razpršenost točk skupine $W = 13$, vendar lahko pri večini podskupin potrdimo konvergenco.

Največja variabilnost je pri skupini $W = 9$, ki nastopi pri višjih stopnjah variabilnosti $dv$. Manj so motivi ohranjeni, slabše bo utežena matrika $Q$, kar pa vpliva na smer Markovske verige. Podoben vpliv pa ima tudi dolžina motiva, saj s krajšo dolžino slabše utežimo matriko $Q$. Kadar pa je dolžina zelo kratka, kot na primer $W = 6$ pa je algoritem velikokrat našel rešitev že v prvem delu, ki je boljša od rešitve drugega dela ali pa je Gibbsovega vzorčevalnika ni niti zagnal. V primerih, kjer pa je rešitev določena s prvim delom, pa je Gibbsov vzorčevalnik imel koeficient $R$ blizu ena, kar nakazuje na konvergenco. Sledje lahko pripišemo k dobro določeni začetni točki, ki jo algoritem deloma določi v prvem delu.

### 5.2.2 Realni podatki

Realne podatkovne množice smo dobili iz prosto dostopne baze, ki so jo zgradili Tompa in sodelavci [48]. Na teh podatkovnih množicah so v svojem članku predstavili rezultate statistične primerjave uspešnosti 13 različnih algoritmov za iskanje motivov v DNA sekvencah. Realne sekvence pripadajo štirim vrstam organizmov, in sicer glive, muhe, miši in človeka. Njihove osnovne karakteristike so podane v tabeli 4.3. V prilogi B je predstavljena primerjava uspešnosti trinajstih algoritmov in algoritma *GraphGibbs* na manjšem izbranem vzorcu, na katerem je algoritem *GraphGibbs* podal boljše rezultate od večine algoritmov na množicah sekvenc človeka, miši in muhe. Algoritem *GraphGibbs* je peta najboljša metoda med štirinajstimi glede na splošno uspešnost na celotnem vzorcu.

Algoritem smo na vsaki množici pognali desetkrat in vzeli povprečja kumulativne vsote pozitivnih, negativnih, lažno pozitivnih in lažno negativnih zadetkov od desetih poskusov. Za grafično analizo posameznih statistik smo vzeli utežena povprečja rezultatov glede na vrsto organizma in jih prenesli na polarne grafikone za primerjavo statistik med posameznimi vrstami algoritma. Kot pri generiranih podatkovnih množicah smo tudi pri realnih podatkih obdržali samo dva prosta parametra, in sicer dolžino motiva in število pojavitev motiva.

Za razliko od generiranih množic imamo pri realnih manjšo kontrolo nad obliko in topologijo posameznih vzorčnih nizov v znani vzorčni množici. Vzorčni nizi so pri realnih sekvencah določeni eksperimentalno in imajo različne dolžine. Prav tako imajo pri danih množicah, ki smo jih uporabili pri naši analizi, motivi visoko stopnjo variabilnosti, kar implicira, da množica vsebuje dva ali več različnih motivov. Interpretacija funkcionalnosti motivov v teh množicah je izven obsega našega raziskovanja, zato smo se odločili, da vključimo vse pojavitve motiva pod en tip, za katerega predpostavimo, da ima visoko stopnjo variabilnosti. Za pričakovano število pojavitev motiva vzamemo torej kar vse znane pojavitve v množici.

Realni vzorčni nizi so različnih dolžin in v množici se lahko pojavijo velike razlike med najdaljšim in najkrajšim vzorčnim nizov. Zato smo za določitev dolžine motiva vzeli mediano dolžin vseh vzorčnih nizov. Mediana je bolj robustna mera lokacije v primerjavi s povprečjem, saj ni občutljiva na ekstremne vrednosti. Algoritem privzame enotno dolžino za vse vzročne nize, zato izbira mediane preceni ali podceni napovedno vrednost algoritma predvsem na nukleotidnem nivoju.

Preverjanje algoritma na realnih podatkih je manj kontrolirano, kljub predpostavkam, ki smo jih privzeli. V ta namen smo realne množice testirali s štirimi različnimi kombinacijami dveh prostih parametrov, in sicer:

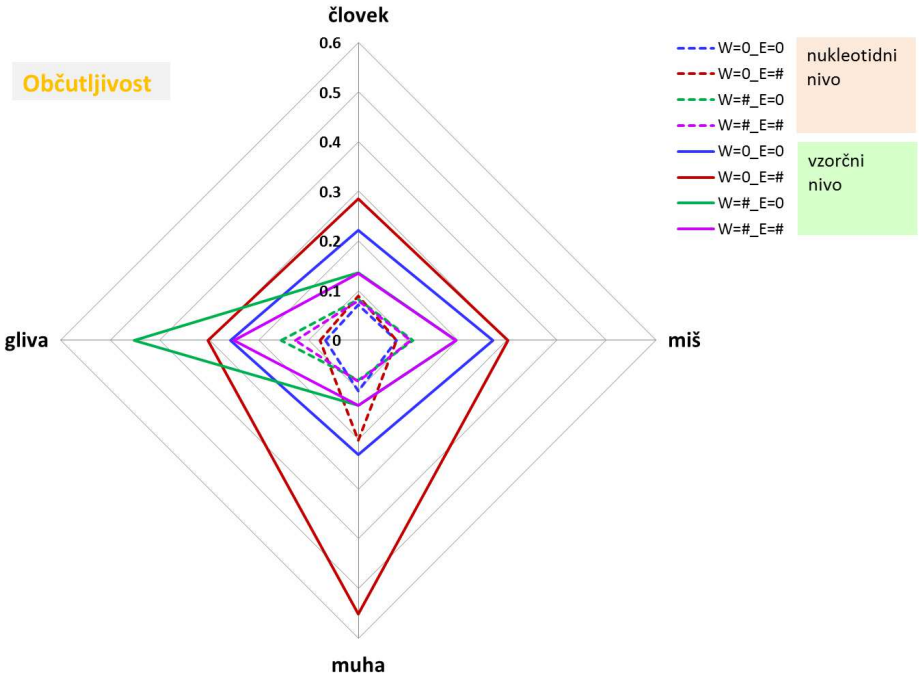
1. dolžina motiva in število pričakovanih pojavitev nista določena:
   $W = 0$ & $E = 0$;

2. dolžina motiva ni določena in število pričakovanih pojavitev je določeno:
   $W = 0$ & $E = \sharp$;

3. dolžina motiva je določena in število pričakovanih pojavitev ni določeno: $W = \sharp$ & $E = 0$;

4. dolžina motiva in število pričakovanih pojavitev sta določena: $W = \sharp$ & $E = \sharp$.

Na sliki 5.19 je polarni graf za vse štiri vrste organizmov. Vrednosti občutljivosti algoritma na realne množice so prikazane na obeh nivojih in za vse štiri kombinacije hkrati. Na vzorčnem nivoju so črte polne, na nukleotidnem nivoju pa so črte črtkane. Na obeh nivojih ima vsaka kombinacija algoritma svojo barvo.



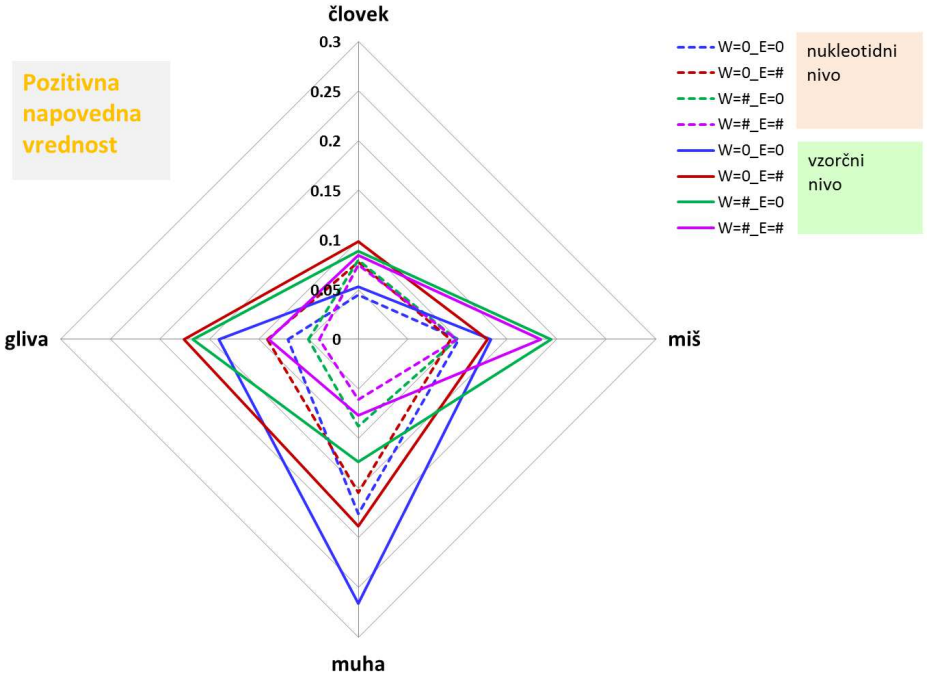

Slika 5.19: Polarni graf občutljivosti algoritma *GraphGibbs* na realnih množicah pri štirih nastavitvah algoritma in primerjava posameznih vrst organizma na obeh nivojih.

Izkazalo se je, da je algoritem najbolj občutljiv na obeh nivojih pri drugi nastavitvi, kjer dolžino motiva določi algoritem in število pričakovanih pojavitev motiva določi uporabnik. Najbolj občutljiv je algoritem na množice DNA sekvenc muhe. Prva nastavitev je podala približno enako stopnjo občutljivosti pri vseh vrstah; marginalno je najvišja občutljivost pri miši. Pri tretji nastavitvi izstopajo rezultati pri množicah za glive. Najnižje vrednosti so pri četrti nastavitvi, kjer smo določili tako dolžino motiva kot tudi pričakovano število pojavitev.

Stopnja občutljivosti je nižja na realnih podatkih nasproti generiranih podatkov, zaradi višje variabilnosti osnovnih lastnosti vzorčnih nizov. Prav tako je najnižja pri vseh nastavitvah pri množicah DNA sekvenc človeka. Človeške sekvence so med vrstami tudi najbolj kompleksne in imajo največ "šuma" med danimi vrstami, kar je posledica evolucije [51]. Po drugi strani pa imamo sekvence gliv, ki so bolj enostavne, na katerih je pri treh nastavitvah občutljivost najvišja na obeh nivojih. Izjema je

druga nastavitev, kjer je občutljivost pri muhi višja od 0.5, predvsem na vzorčnem nivoju.

Oblike grafikona posamezne nastavitve na nukleotidnem nivoju so podobne oblikam grafikonov na vzorčni ravni, a so na manjši skali. To nakazuje, da manjši delež najdenih vzorčnih nizov na nukleotidni ravni pokriva znane vzorčne nize. To posledico prepišemo predvsem variabilnosti dolžin posameznih vzorčnih nizov na eni strani in uniformno (pogosto majhno) dolžino najdenih vzorčnih nizov na drugi strani. Najboljši rezultati na nukleotidnem nivoju so za prvo in drugo nastavitev pri množicah muhe, medtem ko so za drugi dve nastavitvi najobetavnejši rezultati pri množicah gliv.




Slika 5.20: Polarni graf pozitivne napovedne vrednosti algoritma *GraphGibbs* na realnih množicah pri štirih nastavitvah algoritma in primerjava posameznih vrst organizma na obeh nivojih.

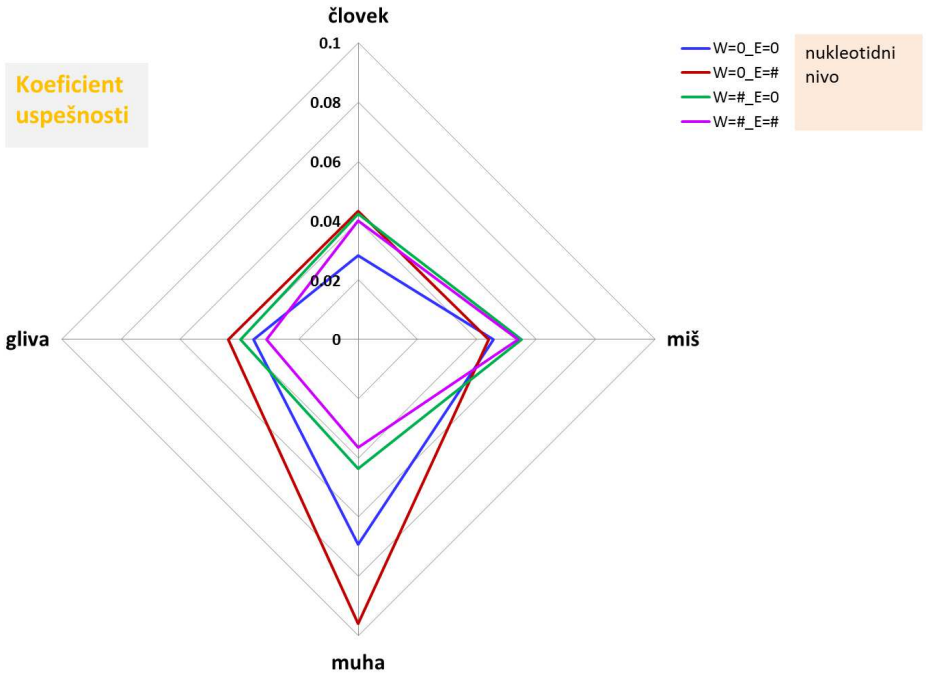
Podobno kot za občutljivost lahko naredimo analizo pozitivne napovedne vrednosti na obeh nivojih in s primerjavo štirih skupin in štirih nastavitev algoritma. To analizo grafično prikažemo s polarnim grafom na sliki 5.20. Skala grafa je manjša od skale na sliki 5.19, kar nakazuje, da je delež najdenih vzorčnih nizov, ki so tudi znani, večji kot delež znanih vzorčnih nizov, ki jih je našel algoritem. Večji deleži znanih najdenih vzorčnih nizov so za prvo in drugo nastavitev na obeh nivojih pri množicah sekvenc muhe. Za razliko od grafa 5.19 so pri tretji in četrti nastavitvi najvišji rezultati pri množicah sekvenc miši.

Najvišji rezultat je pri vrsti muhe pri prvi nastavitvi, pri kateri algoritem sam določi dolžino motiva in število pojavitev. Pri ostalih nastavitvah je pričakovano število pojavitev določeno, kar določi najmanjšo velikost vzorčne množice. Kadar v dobljeni

vzorčni množici ni veliko znanih vzorčnih nizov, potem je število lažno pozitivnih zadetkov toliko višje, kar pa vpliva na vrednost pozitivne napovedne vrednosti, ki jo izračunamo po formuli 4.9 na vzorčnem nivoju in po formuli 4.8 na nukleotidnem nivoju.

Na vzorčnem nivoju je višje vrednosti PPV algoritem imel pri tretji nastavitvi pri skupini sekvenc miši in glive. Na nukleotidnem nivoju pa so vrednosti PPV pri tej nastavitvi manjše. Najmanjše vrednosti pri vseh nastavitvah algoritma so pri človeških sekvencah, kjer so vzorčni nizi precej razpršeni glede na dolžino in stopnjo variabilnosti.

Na nukleotidnem nivoju lahko preverimo tudi koeficient uspešnosti algoritma in korelacijski koeficient med najdeno in znano poravnavo. Na sliki 5.21 je polarni grafikon koeficienta uspešnosti pri posameznih skupinah množic in pri vseh štirih nastavitvah. Oblika grafikona pri posameznih nastavitvah spominja na obliko grafikona 5.20, vendar na manjši skali. Koeficient upešnosti predstavlja delež resnično pozitivnih zadetkov algoritma glede na število vseh možnih zadetkov. Večje število lažno pozitivnih zadetkov predvsem na nukleotidnem nivoju sorazmerno zmanjša koeficient uspešnosti.

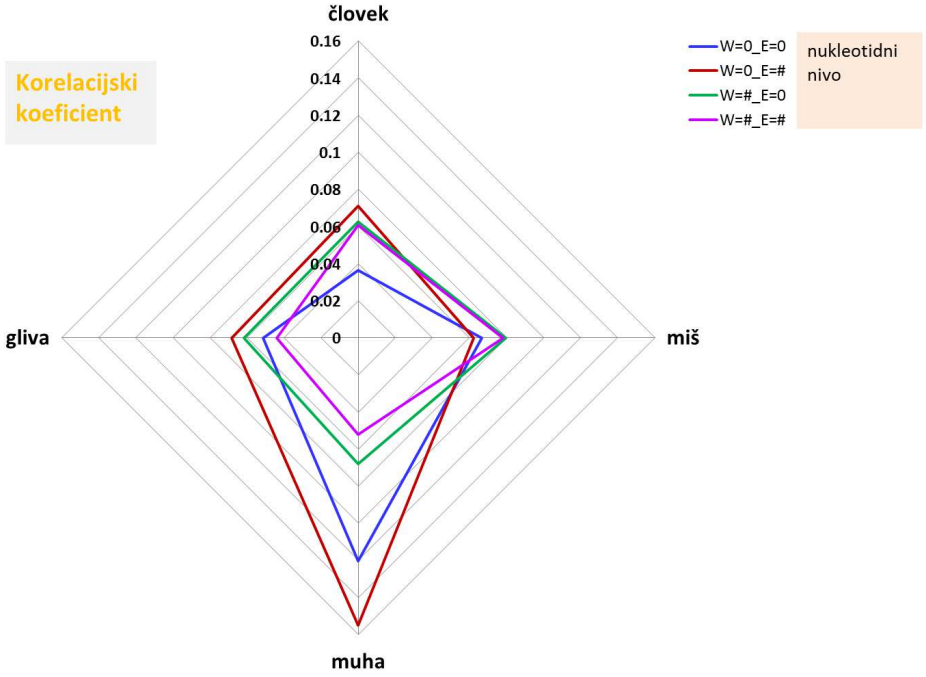



Slika 5.21: Polarni graf koeficienta uspešnosti na nukleotidnem nivoju algoritma *GraphGibbs* na štirih skupinah realnih množic pri štirih nastavitvah algoritma.

Pri skupini muha je vrednost koeficienta uspešnosti najvišja pri prvih dveh nastavitvah. Pri prvi nastavitvi je algoritem dosegel najvišjo vrednost še pri skupinah gliva in človek. Tretja nastavitev je podala višje rezultate pri skupinah gliva, človek in miš, medtem ko je imel algoritem pri skupini muha drugo najvišjo vrednost

koeficienta uspešnosti pri drugi nastavitvi. Razen pri mišjih sekvencah ima algoritem najslabše rezultate pri četrti nastavitvi, kjer sta oba prosta parametra določena.

Na sliki 5.22 je prikazan polarni grafikon za korelacijski koeficient za vse štiri skupine in vse štiri nastavitve algoritma *GraphGibbs*. Pri prvi nastavitvi algoritma so najboljši rezultati za vse štiri skupine razen miši. Pri skupini muha je vrednost korelacijskega koeficienta celo najvišja in relativno na ostale skupine najvišja še pri drugi nastavitvi. Zadnji nastavitvi sta podali najboljše rezultate na mišjih DNA sekvencah.

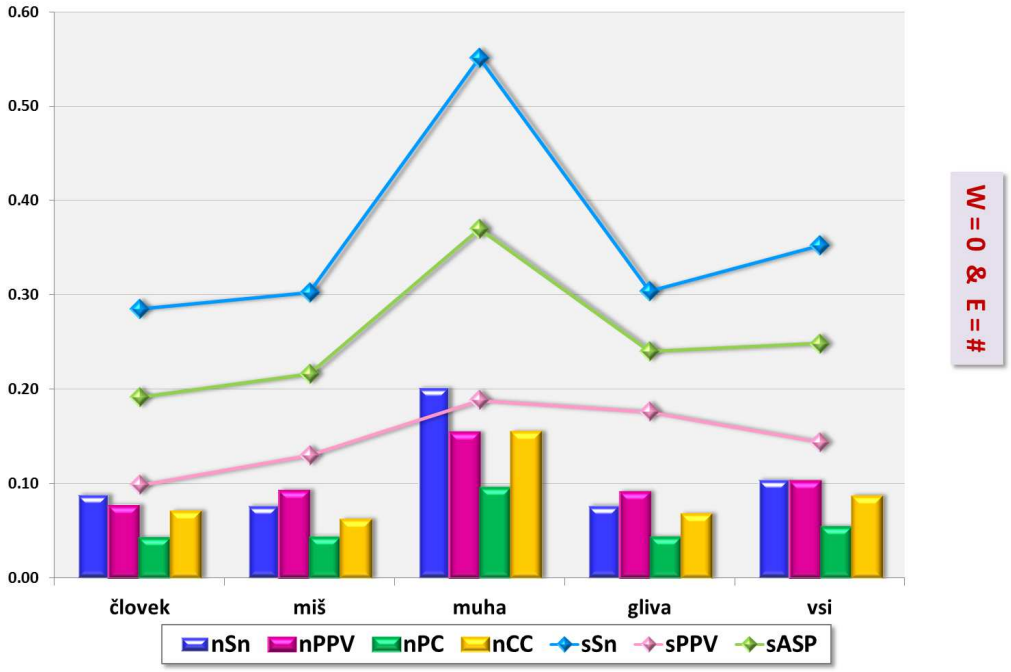



Slika 5.22: Polarni graf korelacijskega koeficienta na nukleotidnem nivoju algoritma *GraphGibbs* na štirih skupinah realnih množic pri štirih nastavitvah algoritma.

Na nukleotidnem nivoju smo preverili še specifičnost algoritma *GraphGibbs* na realnih množicah. Polarni graf na sliki C.9 prikazuje, kako dobro je naš algoritem klasificiral ozadje kot ozadje. Za uspešno delovanje algoritma mora biti ta klasifikacija čim bolj natančna in posledično delež čim bližje 1. Najbližje ena so vrednosti pri drugi nastavitvi za skupini človeka in gliv. Za marginalno razliko je vrednost pri drugi nastavitvi manjša od prve. Pri slednji se je pri DNA sekvencah muhe izkazalo, da je takšna nastavitev bolj učinkovita pri klasifikaciji ozadja. Razen za skupino množic glive pri ostalih skupinah ni prevelik upad ostalih dveh oziroma treh nastavitev. Pri sekvencah gliv pa sta se za odtenek slabše izkazali zadnji dve nastavitvi.

Najvišje vrednosti statistik za merjenje uspešnosti je pri večini realnih množic algoritem dosegel pri drugi nastavitvi, kjer je število pojavitev motiva določeno in dolžina motiva prepuščena optimalni izbiri algoritma. Na sliki 5.23 je prikazana primerjava

vseh statistik za merjenje uspešnosti tako na vzorčnem kot na nukleotidnem nivoju pri vseh štirih skupinah realnih množic pri drugi nastavitvi prostih parametrov. Dodatno je prikazana vrednost statistik na kumulativnih vrednostih pozitivnih in negativnih zadetkov ter lažno pozitivnih in lažno negativnih zadetkov vseh realnih množic, da predstavimo splošno uspešnost algoritma na realnih množicah.




Slika 5.23: Primerjava uspešnosti algoritma *GraphGibbs* pri drugi nastavitvi prostih parametrov z vrednostmi statistik nSn, nPPV, nPC in nCC na nukleotidnem nivoju (stolpci) ter vrednosti statistik sSn, sPPV in sASP na vzorčnem nivoju (črte).

Z grafa 5.23 vidimo, da je algoritem *GraphGibbs* najbolj občutljiv na DNA sekvence muhe na obeh nivojih. Najmanjša natančnost algoritma pa se je izkazala pri sekvencah človeka. Razlika med občutljivostjo in pozitivno napovedno vrednostjo na vzorčni ravni nakazuje, da algoritem določi več lažno pozitivnih pojavitev motiva kot lažno negativnih. Torej je presek med znano in napovedano vzorčno množico velik, vendar je moč napovedane vzorčne množice večja od znane vzorčne množice. To vpliva tudi na koeficient uspešnosti na nukleotidni ravni, ki je najnižji od prikazanih statistik. Korelacija med znano in napovedano poravnavo je pozitivna, kar pomeni, da je vsaj en del teh dveh poravnav sovpada.

Podobni grafi za ostale tri nastavitve so podani v prilogi C.3. Na sliki C.10 je v nasprotju z rezultati pri drugi nastavitvi, najnižja občutljivost algoritma na vzorčnem nivoju pri množicah sekvenc muhe. Po drugi strani pa je pozitivna napovedna vrednost za te množice višja kot pri drugi nastavitvi in vidno višja nasproti ostalih skupin. Pri tretji nastavitvi, katere rezultati so prikazani na sliki 6.3, je algoritem najbolj občutljiv na množice gliv. Marginalno je povišana občutljivost in pozitivna napovedna vrednost pri sekvencah miši. Četrta nastavitev pa v povprečju najbolj

ustreza za obdelavo množic miši, saj so tukaj vse vrednosti statistik najvišje z izjemo občutljivosti na obeh nivojih pri množicah gliv.

# 6. Razprava

V prejšnem poglavju smo prikazali učinkovitost delovanja algoritma *GraphGibbs* na generiranih in realnih podatkovnih množicah. S Plackett-Burmanovim eksperimentalnim načrtom smo preverili, kateri parametri algoritma najbolj vplivajo na dobljeno rešitev. Na iskanje rešitve pa vplivajo tudi predpostavke, s katerimi smo oblikovali algoritem *GraphGibbs*.

## 6.1 Osnovne predpostavke

Nekaj osnovnih predpostavk, ki smo jih uporabili pri razvijanju metode in generiranju množic, in sicer:

- da je v vsaki sekvenci v množici vsaj ena pojavitev motiva;
- da so vsi vzorčni nizi enake dolžine;
- da ima določen vzorec vsaj $\lfloor N/2 \rfloor$ pojavitev v množici podatkov, pri čemer je $N$ število sekvenc v dani množici in
- da je stopnja variabilnosti v vzorčnih množicah $dv$ največ $W/4$, kjer je $W$ dolžina vzočnih nizov.

Prva predpostavka predvideva homogenost množice oziroma, da DNA sekvence pripadajo isti vrsti organizma in so odseki primerno izbrani. To pa ne drži vedno pri realnih podatkovnih množicah, kar se je izkazalo tudi pri realnih množicah podatkov uporabljenih v tem delu. V teh primerih takšna omejitev preceni število pojavitev motiva in otežuje konvergenco Gibbsovega vzorčevalnika. Markovska veriga, ki jo gradimo pri Gibbsovem vzorčevalniku, je pod vplivom pozicijsko utežene matrike $\mathcal{Q}$ in vektorja ozadja $\mathcal{P}$. Obe strukturi pa sta odvisni od števila vzorčnih nizov v trenutni vzorčni množici in vplivata na vrednost statistike $I$.

Kadar je vzorčna množica preveč heterogena oziroma so pripadajoči vzorčni nizi preveč variabilni, potem se to odraža na matriki $\mathcal{Q}$. Takšno razpršenost v zapisu vzorčnih nizov pa lahko, med drugim, dobimo kot posledico prve predpostavke. Po njej namreč vključimo v vzorčno množico tudi segmente sekvenc, v katerih dejansko ni odgovarjajočega znanega vzorčnega niza. Z izjemo slučajne izbire segmenta, ki ima zelo podoben zapis kot ˝pravi˝ vzorčni nizi, bo vključitev dodatnih vzorčnih nizov za zadostitev predpostavke ovirala konvergenco Gibbsovega vzorčenja.

Z drugo predpostavko smo poenotili dolžino motiva na vse vzorčne nize v množici. Pri realnih množicah, na katerih smo preverili uspešnost našega algoritma, dolžine vzorčnih nizov niso enotne. Med drugim je to posledica izbora sekvenc iz TRANSFAC baze, v kateri so motivi nekarakteristično dolgi. Tompa s sodelavci, [48], je zgradil podatkovno zbirko samo iz TRANSFAC baze. Dobljene množice imajo vzorčne nize različnih dolžin, ki se lahko razlikujejo tudi do 40 nukleotidov v dolžini zapisa motiva.

Brez poenotenja dolžine vseh vzorčnih nizov je problem precej kompleksnejši. Spomnimo se, da v praksi raziskovalci iščejo *neznane* motive. S tem niso samo neznani položaji posameznih vzorčnih nizov, ampak tudi njihove dolžine. Ker ne poznamo pravih dolžin, se pri modeliranju omejimo na interval možnih dolžin za vrednost $W$, ki pa je enaka vsem vzorčnim nizom. Enotna dolžina pa lahko nekatere "prave" vzorčne nize skrajša ali pa podaljša, kar se odrazi na občutljivosti in pozitivni napovedni vrednosti algoritma na nukleotidnem nivoju. Pri osnovnem algoritmu so avtorji, [27], omilili omejitve te predpostavke, tako da so podali možnost primerjave poravnave motivov z različnimi enotnimi dolžinami. Osnovni algoritem lahko tudi išče rešitev pri različnih vrednostih za parameter $W$ in za končno rešitev vzame tisto dolžino motiva, katere poravnava maksimizira vrednost statistike $I$.

Pri algoritmu *GraphGibbs* ima druga predpostavka podobne posledice kot pri osnovnem algoritmu, torej lahko podceni ali preceni dolžino določenih pravih vzorčnih nizov. Kot pri osnovnem algoritmu lahko tudi naš algoritem določi optimalno vrednost za $W$ iz intervala možnih dolžin, ki jih poda uporabnik. Dodatno pa je omogočena možnost, da algoritem sam izbere optimalno dolžino iz intervala $[6, 30]$. V prvem primeru je kot pri osnovnem algoritmu dolžina vzorčnih nizov končne rešitve določena s poravnavo motiva, ki da največjo vrednost statistike $I$. V drugem primeru pa je dolžina vzorčnih nizov določena z dolžino popolnoma ohranjenega motiva, ki se v množici pojavi dovolj pogosto. Popolnoma ohranjeni nizi pa so večinoma kratki in posledično omejijo občutljivost algoritma *GraphGibbs* na nukleotidnem nivoju, saj podcenijo prave dolžine vzorčnih nizov.

S tretjo predpostavko postavimo spodnjo mejo za število pojavitev nekega niza, da ga lahko obravnavamo kot vzorec. Spodnja meja zahteva pojavitev vzorca v približno polovici danih sekvenc, drugače lahko smatramo, da je dana množica podatkov preveč heterogena oziroma ne vsebuje segmentov DNA sekvenc, ki imajo isto funkcionalnost v DNA zapisu. Z večanjem števila sekvenc $N$ v množici se spodnja meja tudi veča, kar pa vpliva na iskanje vzorcev v prvem delu algoritma *GraphGibbs*. Večja spodnja meja pomeni krajšo dolžino vzorca v prvem delu, saj so pripadajoči vzorčni nizi popolnoma ohranjeni. Pri realnih množicah pa imajo motivi pogosto (visoko) stopnjo variabilnosti, kar pomeni, da je malo identičnih vzorčnih nizov znotraj vzorčne množice. S tretjo predpostavko pa predpostavimo, da je vsaj polovica vzorčnih nizov identičnih.

Poravnava najdenega vzorca v prvem delu algoritma *GraphGibbs* lahko zaradi upoštevanja tretje predpostavke začne Gibbsovo vzorčenje na "napačnem" delu prostora

rešitev. Besedo napačno smo dali v navednice, saj je vsak vzorec določen v prvem delu statistično nadvpovprečno zastopan v dani množici ravno zaradi dane spodnje meje. Večje število pojavitev v množici kot bi jih bilo slučajno nakazuje, da ima dobljen vzorec lahko neko funkcijo v DNA sekvenci. Vzorčna množica v prvem delu je iz tega vidika lahko vedno relevantna, čeprav ne privede nujno zmeraj do prave rešitve.

Zadnja izmed predpostavk določa sprejemljivo stopnjo variabilnosti, kot smo jo definirali z definicijo 1.2.4 in smo jo uporabili primarno pri generaciji množic. Vzorec je relativno ohranjen, ko je stopnja variabilnosti primerno nizka. Pri razvijanju metode smo postavili to mejo na četrtino pričakovane dolžine motiva, ki smo se je držali tudi pri generiranju množic za testiranje. Rezultati algoritma *GraphGibbs* na teh množicah so pokazali, da se z večanjem variabilnosti vzorčnih nizov manjša učinkovitost zaznave znane vzorčne množice. To je pričakovan izid, saj se variabilnost v zapisih vzorčnih nizov odraža v matriki $\mathcal{Q}$ in posledično statistiko $I$, s katero ocenjujemo poravnavo najdenega motiva.

## 6.2 Modeliranje z grafom

S prvim delom algoritma *GraphGibbs* analiziramo DNA sekvence z modelom grafa. Na podlagi genskega koda, tabela 1.1, preberemo zaporedje trojčkov v sekvencah in te povezave prestavimo v obliki usmerjenega multigrafa. Lastnosti multigrafa pomagajo določiti število vzorčnih množic, njihovo velikost in dolžino njihovih presečnih vzorcev. Pomembno pa je upoštevati, da so najdeni vzorčni nizi popolnoma ohranjeni, kar pogosto pomeni, da so zelo kratki. V primeru, ko dolžina motiva ni določena s strani uporabnika, bo tudi motiv, ki predstavlja rešitev, imel kratko dolžino, saj algoritem v drugem delu določa samo vzorčne nize in ne več števila ter lastnosti vzorčnih množic.

Kratke dolžine motiva v prvem delu lahko podcenijo pravo dolžino motiva. Posledica tega je nižja občutljivost algoritma na nukleotidnem nivoju. Določanje števila pojavitev motiva v vzorčni množici je odvisno od števila pojavitev popolnoma ohranjenega motiva. Kadar dolžina motiva ni določena vnaprej, potem algoritem v prvem delu poišče popolnoma ohranjen motiv najdaljše dolžine, tako da je število pojavitev vsaj polovica števila sekvenc oziroma $E \geq \lfloor N/2 \rfloor$. Slednji pogoj je omilitev emirične predpostavke $E \geq N$, ki jo upoštevajo mnoge metode za iskanje motivov; med drugimi osnovni algoritem, EM algoritem iz tabele 1.4. Ta predpostavka je pa splošna predpostavka pri problemih prepoznavanja vzorcev. Namreč narava vzorca diktira dovolj pogosto število pojavitev v sekvencah, tekstu ali poljubni množici podatkov.

Pri iskanju motivov v DNA sekvencah je število pojavitev $\psi(v)$ popolnoma ohranjenega vzorca $v$ v obratnem razmerju z dolžino vzorca. To pomeni, da se z večanjem dolžine motiva manjša število pojavitev, kar pa lahko podceni dejansko število pojavitev motiva. Zato v prvem delu poiščemo krajši podniz $v_{kratek}$ v vzorcu $v$, ki ima večje število pojavitev, to je $\psi(v_{kratek}) > \psi(v)$. Kadar najdemo ustrezen pod-

niz nadaljujemo z njegovo poravnavo, vendar z dolžino $\ell_v$ vzorca $v$ oziroma z dano dolžino $W$. Podniz pa ne sme biti prekratek, saj bi potem algoritem lahko precenil število pojavitev, kar zniža pozitivno napovedno vrednost algoritma. Ko je število pojavitev predvideno oziroma je $E$ določen, potem vzamemo najdaljši (popolnoma ohranjen) vzorec $v$, za katerega velja $\psi(v) \leq E$. V primeru, da velja $\psi(v) > E$ je motiv določen in se Gibbsov vzorčevalnik ne požene. Drugače dopolnimo poravnavo $A(v)$ vzorca $v$ s slučajno izbranimi števili in jo določimo kot začetno točko Gibbsovega vzorčevalnika. Manj kot je slučajnih vrednosti, manjša je verjetnost divergence Gibbsovega vzorčevalnika.

Vpliv dolžine motiva $W$ in števila pojavitev $E$ smo potrdili tudi z Plackett-Burman eksperimentalnim načrtom. Oba dejavnika sta pokazala vpliv na izračun statistike $I$, s katero ovrednotimo možno rešitev. Zato je določitev vrednosti teh dveh parametrov v prvem delu pomembna. Kadar sta določena s strani uporabnika, mora algoritem v prvem delu poiskati vzorec, katerega lastnosti se najbolj približajo danim vrednostim $W$ in $E$. Z Gibbsovim vzorčevalnikom nato poiščemo končno poravnavo motiva s temi parametri. Z analizo konvergence, ki smo jo prikazali s sliko 5.18, lahko potrdimo, da v večini primerov s prvim delom zagotovimo konvergenco Gibbsovega vzorčevalnika. Uporabnost končne poravnave, h kateri konvergira Gibbsov vzorčevalnik, pa je prepuščena uporabniku algoritma *GraphGibbs*.

Poglejmo si podrobneje učinek prvega dela algoritma *GraphGibbs* pri reševanju problema. Recimo, da v prvem delu algoritem določi $m$ motivov. Označimo njihove vzorčne množice prvega dela kot $A = \{A_i\}_{i=1}^m$. Vzorčne množice dobljene z vzorčevalnikom pa z $B = \{B_i\}_{i=1}^m$. Velja $|B| \leq |A|$, saj sta lahko presečna vzorca dveh množic iz $B$ enaka, kar pomeni, da sta glede na motiv vzorčni množici enakovredni. Z Gibbsovim vzorčevalnikom smo tako iz dveh različnih začetnih točk prišli do enake oziroma podobne rešitve.

Iz vsake začetne točke, ki jo algoritem določi v začetnem delu, lahko Gibbsov vzorčevalnik pride tudi do različne rešitve. V tem primeru velja $A_i \neq B_i$ za vsak $i = 1, \ldots, m$, torej dobimo $2m$ vzorčnih množic z $2m$ različnimi presečnimi vzorci. Prav tako velja $|A_i| < |B_i|$ za vsak $i$, tako da je $B_i \neq \emptyset$, saj v drugem delu iščemo vzorčne nize, ki spadajo k vzorčni množici $A_i$, vendar imajo višjo stopnjo variabilnosti.

Pri obdelavi rezultatov na generiranih množicah, smo na slikah s krožnimi diagrami 5.17, C.12, C.13, C.14, C.15 in C.16 med drugim prikazali tudi primerjavo med vzorčnimi množicami iz $A$ in množicami iz $B$. Pri nekaterih generiranih množicah s pozitivno stopnjo variabilnosti je algoritem *GraphGibbs* pognal Gibbsov vzorčevalnik, vendar je rešitev prvega dela bolj občutljiva na vzorčni ravni, z boljšo pozitivno napovedno vrednostjo kot rešitev drugega dela algoritma glede na znano vzorčno množico. Iz tega lahko povzamemo, da modeliranje z grafom ni samo korak pri iskanju rešitve, ampak lahko tudi poda samostojno rešitev, ki bolje zazna poravnavo znanega motiva.

Po gornjih predpostavkah algoritem *GraphGibbs* v prvem delu določi vzorec z do-

volj velikim številom pojavitev. Kot smo že omenili, takšno število pojavitev lahko nakazuje pomembno vzorčno množico, ampak ne vemo ali je najden motiv za raziskovalca nezanimiv in ga lahko filtriramo, ali pa dejansko predstavlja funkcionalen motiv. Četudi se v primeru realnih množic dobljeni rezultati težje klasificirajo, bo naš algoritem, pod splošnimi pogoji, navedel vse nadpovprečno zastopane vzorce, ki jih najde v množici. Takšni vzorci pa zahtevajo podrobnejšo obravnavo glede njihove funkcije in naloge v sekvenci. Te ugotovitve pa prepustimo uporabniku algoritma *GraphGibbs*.

Poleg modela grafa smo osnovnemu algoritmu dodali še prirejeno porazdelitveno funkcijo $\Gamma$, da smo v model lahko vključili možnost neenakomerno porazdeljenih pojavitev motiva. Po prvi predpostavki še vedno privzamemo, da je vsaj ena pojavitev v vsaki sekvenci. Njen vpliv na rezultat smo že omenili, vendar vpliva tudi na določanje porazdelitvene funkcija $\Gamma$ in posledično na vektor pričakovanih pojavitev motiva na sekvenco $\boldsymbol{\xi}$. Po predpostavki so vse vrednosti v $\boldsymbol{\xi}$ pozitivne. Za določanje števila pojavitev smo uporabili dodatne informacije o strukturi sekvenc iz prvega dela. S pomočjo poravnave $A(v)$ popolnoma ohranjenega vzorca $v$ določimo začetno porazdelitev pričakovanih števil kot tudi hiperparametra za Beta porazdelitev, po kateri vzorčnimo vrednosti $P(E_i = c)$, kjer je $E_i$ število pričakovanih pojavitev v sekvenci $S_i$ in $c = 1, \ldots, C$, za $C = \max\{\psi_j(v)\}_{j=1}^{N}$ oziroma največje število najdenih pojavitev vzorca $v$ po vseh sekvencah.

Dodatna informacija, ki jo dobimo po analizi multigrafa v prvem delu, nam omogoča bolj natančno določanje porazdelitvene funkcije $\Gamma$. Pri realnih podatkih, ki so nam bili na voljo, pa smo lahko ugotovili, da obstajajo sekvence v množici, ki ne vsebujejo nobenega znanega vzorčnega niza. Iskanje porazdelitve vzorčnih nizov bi morali prirediti, da upoštevajo še slednjo možnost, kjer sekvenca ne vsebuje pojavitve motiva. Kako določiti verjetnost $P(E_i = 0)$ za neko sekvenco $S_i$ pa predstavlja poseben izziv, ki ga je potrebno še raziskati. Z rešitvijo tega problema lahko izboljšamo delovanje algoritma *GraphGibbs* na realnih množicah podatkov.

Določanje pričakovanega števila pojavitev motiva na sekvenco, poleg poravnave popolnoma ohranjenga vzorca, ki smo ga našli s prvim delom, vpliva tako na hitrost konvergence Gibbsovega vzorčevalnika kot tudi na njegovo smer iskanja. Pri večini generiranih množic je algoritem *GraphGibbs* z Gibbsovem vzorčevalnikom dosegel stacionarno verigo oziroma poravnavo. Pri realnih množicah je pa potrebna podrobnejša analiza konvergence za določitev primernega števila iteracij.

Pri vsaki iteraciji se poravnava oceni s statistiko $I$, ki pa je občutljiva na dolžino motiva in spremembe v številu vzorčnih nizov. Dolžina motiva in število vzorčnih nizov sta pri *GraphGibbs* povezana v prvem delu algoritma, kjer iščemo najdaljši popolnoma ohranjen vzorec, ki ima dovolj veliko število pojavitev. Na število vzorčnih nizov v vzročni množici rešitve vpliva tudi prva predpostavka. Pri našem algoritmu se je tako statistika $I$ izkazala za preveč občutljivo statistiko, predvsem pri množicah s kratkimi motivi in pri množicah z neenakomerno porazdeljenimi pojavitvami motiva.

Podobna ideja za uporabo grafičnega modela je bila implementirana z uporabo klik. V članku [43] so obravnavali utežene klike v grafu, ki so ga zgradili iz vhodnih sekvenc. Vsako ogljišče v njihovem grafu predstavlja podzaporedje dolžine $W$ in povezava so samo med ogljišči, katerih ocena poravnave doseže določen prag. Vendar menimo, da je bolj intuitivno modelirati ohranjene vzorce namesto klik, saj tako zrcalimo notranjo strukturo vhodnih sekvenc.

## 6.3 Rezultati

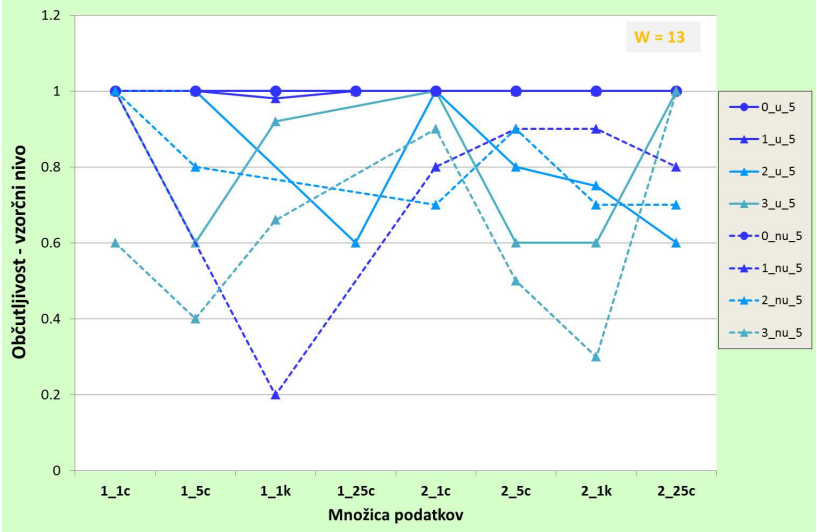
Krajši opis in razlaga modeliranja DNA sekvenc z grafom je opisana tudi v našem članku [44]. V članku smo objavili še primerjavo rezultatov *osnovnega algoritma* in algoritma *GraphGibbs* na manjšem vzorcu generiranih množic. Primerjava je nakazala, da algoritem *GraphGibbs* natančneje določi znano poravnavo. Prav tako je občutljivost algoritma *GraphGibbs* višja od občutljivosti *osnovnega algoritma*, vendar ima naš algoritem večjo časovno zahtevnost. Potek algoritma *GraphGibbs* je pričakovano daljši, saj smo *osnovnemu algoritmu* dodali še modeliranje DNA sekvenc. Povečanje časovne zahtevnosti pa je zanemarljivo majhno, medtem ko je občutljivost in natančnost opazneje boljša.

### Generirane množice z enim motivom

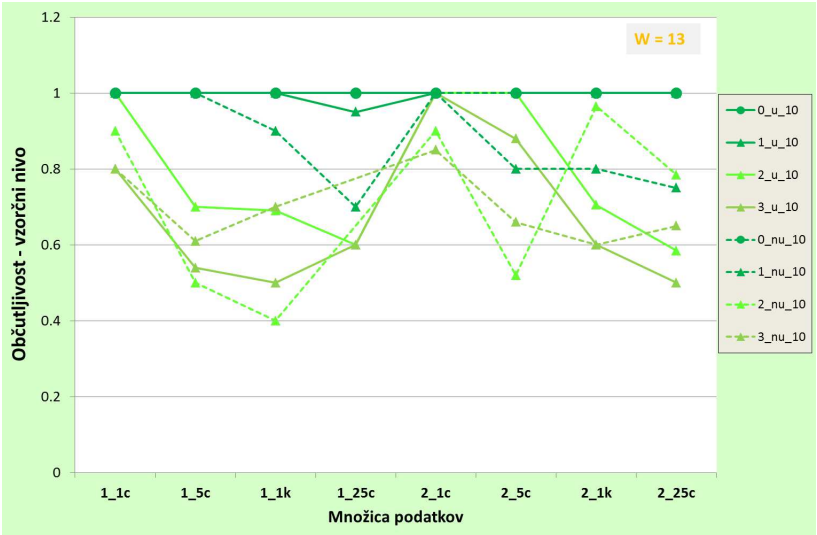
Število sekvenc v množici in dolžina motiva vpliva na iskanje rešitve pri našem algoritmu. To se je pokazalo tudi pri predstavitvi rezultatov algoritma na generiranih množicah. Iz primerjalnih grafov 5.6, 5.8, 5.10 in 5.11 lahko razberemo, da ima algoritem v povprečju najmanjšo razpršenost rezultatov pri skupinah z desetimi sekvencami. Rezultati posameznih statistik (občutljivost, pozitivna napovedna vrednost, koeficient uspešnosti in korelacijski koeficient) so bili najmanj razpršeni po skupinah z dolžino motiva $W = 13$ in $W = 18$. Poglejmo si skupine z $W = 13$ podrobneje. Na sliki 6.1 je prikazana občutljivost algoritma *GraphGibbs* na vzorčnem nivoju pri posameznih množicah, kjer je $W = 13$.

Ime posamezne množice na sliki 6.1 lahko razberemo preko oznake ($dv\_P\_N$) v legendi in ($E\_L$) na abscisni osi. Oznaka $E$ ima dve vrednosti: 1 in 2, kar predstavlja $N$ in $2N$ pojavitev motiva v množici. Dodatno lahko množice ločimo v tri barvne skupine glede na $N$, in sicer za $N = 5$ modra, $N = 10$ zelena in $N = 15$ rdeča barva. Posamezni odtenki glavne barve dodatno ločijo množice glede na stopnjo variabilnosti $dv$; od najtemnejšega odtenka za popolnoma ohranjene motive ($dv = 0$) do najsvetlejšega pri največji stopnji variabilnosti ($dv = 3$). Polne črte označujejo množice, kjer so pojavitve motiva razporejene enakomerno po sekvencah ($P = u$), črtkane črte pa množice z neenakomerno porazdeljenimi pojavitvami motiva ($P = nu$).

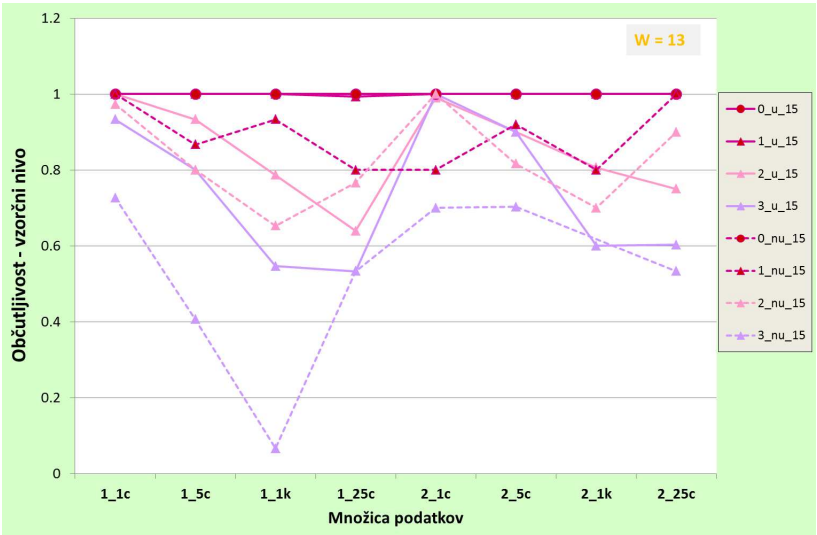
Najmanjša občutljivost algoritma *GraphGibbs* se je pokazala pri dveh množicah, in sicer $13\_3\_nu\_15\_1\_1k$ in $13\_1\_u\_5\_1\_1k$. Pri množicah z $1\_1k$ in $1\_5c$ je tudi največja razpršenost rezultatov algoritma pri merjenju občutljivosti. Kadar imamo vsaj $2N$ pojavitev, je razpšenost rezultatov znotraj in med skupinami precej




(a) Občutljivost na množicah z $N = 5$.




(b) Občutljivost na množicah z $N = 10$.




(c) Občutljivost na množicah z $N = 15$.

Slika 6.1: Občutljivost algoritma *GraphGibbs* na množicah z dolžino motiva $W = 13$.

manjša; tudi pri višjih stopnjah variabilnosti. Več znanih (relativno ohranjenih) vzorčnih nizov je v množici, večjo možnost ima algoritem, da jih zazna. Popolnoma ohranjene motive pa je algoritem pričakovano zaznal v celoti.

Med drugim lahko na sliki 6.1 opazimo, da je najmanjši variacijski razpon v skupini množic z $N = 10$. Prav tako je boljša občutljivost algoritma *GraphGibbs* na množicah, kjer so pojavitve motiva enakomerno porazdeljene v množici. Pri neenakomerni porazdelitvi je zaznava znanih vzorčnih nizov slabša, kar nakazuje na večje število lažno negativnih zadetkov. Torej algoritem ni zaznal vseh znanih vzorčnih nizov, k čemur je prispevala tudi predpostavka o vsaj eni pojavitvi motiva na sekvenco, saj lahko preusmeri iskanje pojavitev motiva v "napačno" smer. Pri generiranju množic z neenakomerno porazdeljenimi pojavitvami motiva smo namreč to predpostavko zanemarili, saj smo želeli simulirati realne podatke. Tako pri takšnih množicah ena ali več sekvenc ne vsebuje nobene pojavitve motiva.

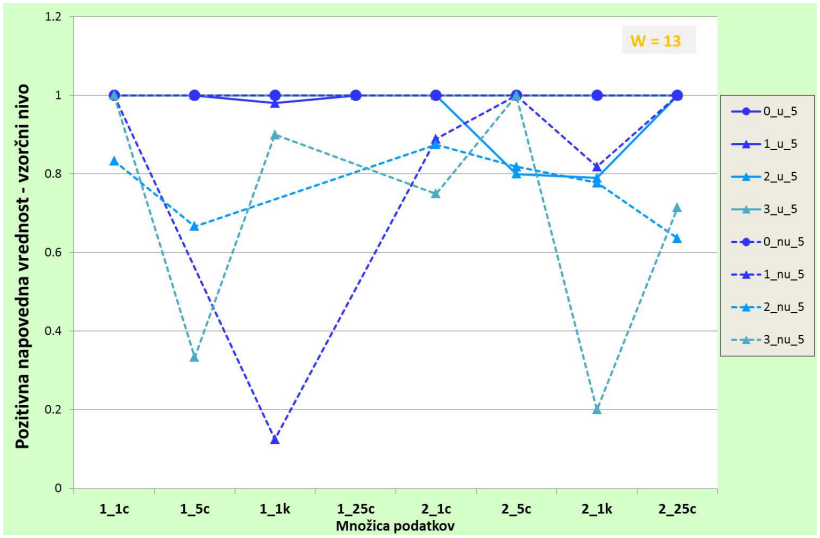
Pri množicah z neenakomerno porazdelitvijo je torej število lažno pozitivnih zadetkov večje. Slednje število pa vpliva na izračun pozitivne napovedne vrednosti. Na sliki 6.2 so prikazani rezultati PPV algoritma *GraphGibbs* na vseh množicah z dolžino motiva $W = 13$. Množice so označene enako kot na sliki 6.1.

Na sliki 6.2 lahko vidimo upad vrednost PPV predvsem na množicah z neenakomerno razporejenimi pojavitvami motiva in tudi pri množicah s sekvencami dolžine $1c$, kjer je bila občutljivost bližje ena. To nakazuje na večje število lažno pozitivnih zadetkov v teh množicah, torej je algoritem poleg znanih vzorčnih nizov, v vzorčno množico določil dodatne (napačne) vzorčne nize.

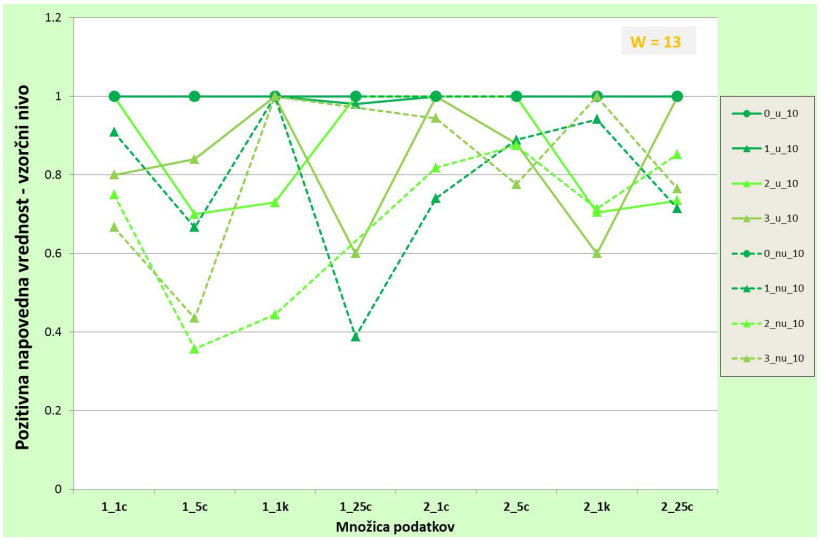
Prav tako si lahko podrobneje pogledamo porazdelitev vrednosti koeficienta uspešnosti in korelacijskega koeficienta na nukleotidnem nivoju na slikah C.25 in C.26 v prilogi C.5. Variacijski razmik je pri koeficientu uspešnosti visok pri večini skupin generiranih množic (glede na abcisno os), saj upošteva tako lažno negativne kot lažno pozitivne zadetke. Korelacijski koeficient je najvišji pri množicah z enakomerno porazdelitvijo pojavitev motiva. Večja razpršenost rezultatov je predvsem pri množicah z vrednostjo $E = 1$ in $N = 5$. Na množicah z večjim številom sekvenc je korelacijski koeficient v povprečju nad vrednostjo 0.5.

V prilogi C.5 se nahajajo še odgovarjajoče slike za preostale dolžine $W \in \{6, 9, 18\}$. Občutljivost in pozitivna napovedna vrednost algoritma *GraphGibbs* na vzorčnem nivoju pri množicah z dolžino motiva $W = 6$ sta prikazana na grafih C.17 in C.18. Koeficient uspešnosti in korelacije na nukleotidnem nivoju pa na odgovarjajočih slikah C.19 in C.20. Občutljivost algoritma se najbolj razlikuje pri množicah s petimi sekvencami. Pri množicah z $dv = 1$ ni opazne razlike med občutljivostjo algoritma, ko so pojavitve motiva enakomerno ali pa neenakomerno porazdeljene.

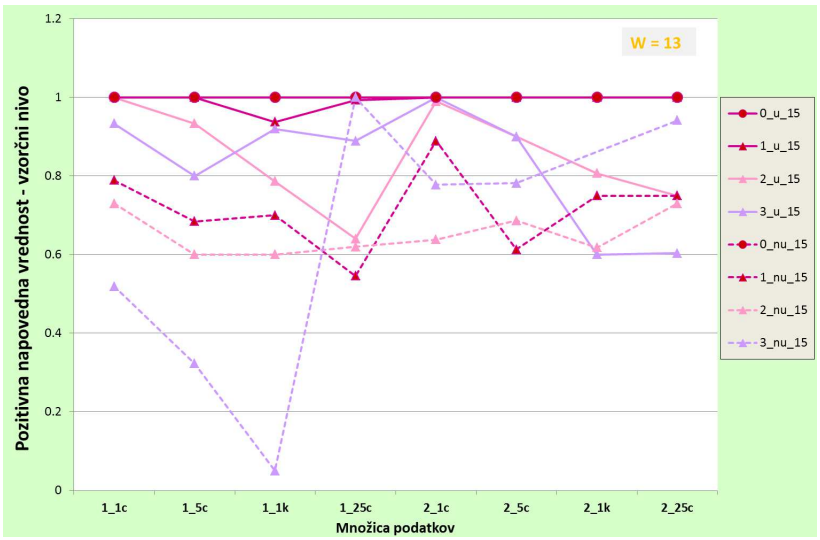
Na sliki C.18 lahko vidimo, da je pri množicah z najkrajšimi motivi ($W = 6$) algoritem *GraphGibbs* pogosto za rezultat podal večjo vzorčno množico kot je znana. Drugače povedano, med rezultati je veliko lažno pozitivnih zadetkov, kar zmanjša pozitivno napovedno vrednost. Pri kratkih motivih algoritem *GraphGibbs* lahko v




(a) Pozitivna napovedna vrednost na množicah z $N = 5$.




(b) Pozitivna napovedna vrednost na množicah z $N = 10$.




(c) Pozitivna napovedna vrednost na množicah z $N = 15$.

Slika 6.2: Pozitivna napovedna vrednost algoritma *GraphGibbs* na množicah z dolžino motiva $W = 13$.

okviru enega pogona zazna znane vzorčne nize pri različnih najdenih vzorčnih množicah. To je posledica prvega dela algoritma, kjer je algoritem samo navzdol omejen z mejo $\lfloor E/2 \rfloor$. Nekateri kratki (nepravi) motivi pa se lahko slučajno pojavijo večkrat kot pričakujemo znotraj množice; posebno pri $E = N$. V prvem delu iščemo ravno motiv z najvišjim številom pojavitev, kar pomeni, da bomo zajeli več pojavitev kot je pričakovano oziroma znano.

Pri daljših motivih se število lažno pozitivnih zadetkov zmanjša, kar se pozna na pozitivni napovedni vrednosti algoritma *GraphGibbs*. Vidna izboljšava v občutljivosti in PPV je že pri dolžini motiva $W = 9$ na odgovarjajočih slikah C.21 in C.22. Pri množicah z $W = 9$ so vrednosti statistik tudi manj razpršene. Na sliki C.21 lahko vidimo, da je algoritem bolj občutljiv na množicah z $P = u$, kadar je pojavitev motiva vsaj $2N$. Pri množicah z $N$ pojavitvami motiva pa ni vidne razlike med množicami z enakomerno in neenakomerno porazdeljenimi pojavitvami motiva.

Za množice z daljšimi motivi ($W = 18$), ki so relativno ohranjeni, je občutljivost in pozitivna napovedna vrednosti algoritma *GraphGibbs* zelo dobra. Iz slik C.27 za občutljivost in C.28 za pozitivno napovedno vrednost na vzorčnem nivoju vidimo, da je algoritem zaznal skoraj vse znane vzorčne nize pri večini generiranih množic in ni določil preveč (dodatnih) lažno pozitivnih vzorčnih nizov. Koeficient uspešnosti, slika C.29, in korelacijski koeficient, slika C.30, na nukleotidnem nivoju prikazujeta, da algoritem ni zaznal nekaterih znanih vzorčnih nizov v celoti predvsem pri množicah z neenakomerno porazdeljenimi pojavitvami motiva. Tukaj spet vidimo vpliv predpostavke o pojavitvi motiva v vsaki DNA sekvenci. Slabša zaznava je izrazitejša pri množicah, kjer je $N = 5$ pojavitev motiva in je dolžina ozadja sekvenc $L = 2500$. Kontrast med uspešnostjo na vzorčnem in na nukleotidnem nivoju nastane predvsem zaradi dolžine motiva $W = 18$. En lažno pozitivni zadetek na vzorčnem nivoju namreč predstavlja 18 lažno pozitivnih zadetkov na nukleotidnem nivoju.

### Generirane množice z dvema motivoma

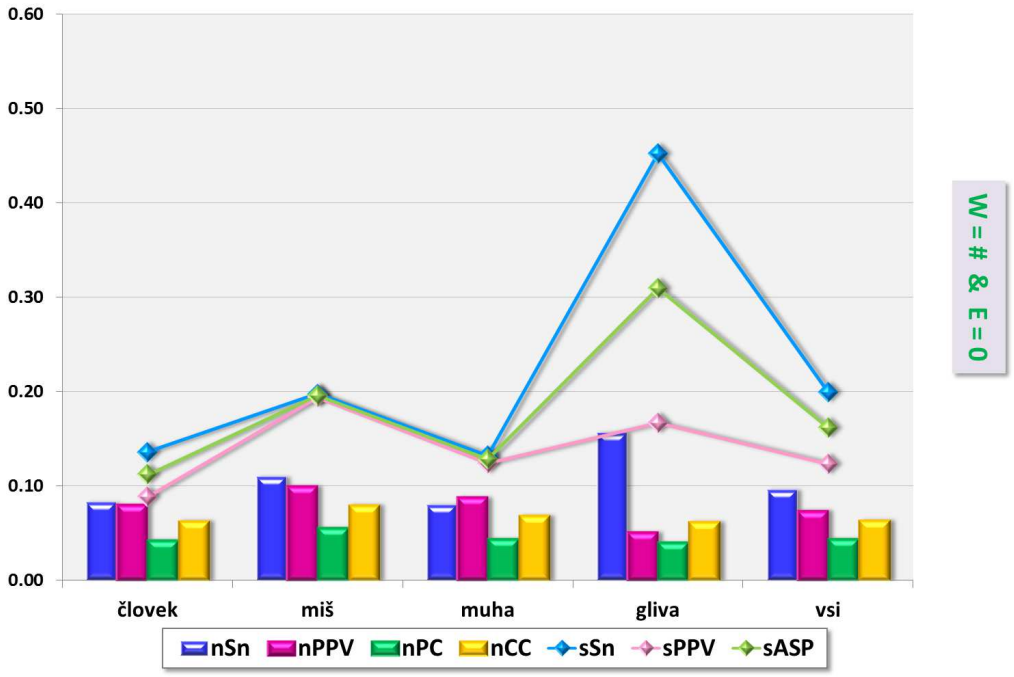
Pri množicah z dvema motivoma sta se izkazala za vplivna dejavnika na rezultat, poleg dolžine in stopnje variabilnosti motiva, tudi zaporedje iskanja posameznega motiva in število pričakovanih pojavitev obeh motivov. Zaporedje iskanja je bilo določeno z vrednostmi za posamezen prosti parameter (to sta bila $W$ in $E$). Na sliki 5.13 lahko opazimo, kako te vrednosti tudi omejijo iskanje motivov. Kadar velja $W_1 < W_2$ in $E_1 \leq E_2$, je večja verjetnost, da algoritem najprej najde motiv $M_2$, a pri pogojih $W_1$ in $E_2$. Ker je $W_1 < W_2$, znani vzorčni nizi motiva $M_2$ ne bodo zaznani v celoti. Na drugi strani pa bo algoritem *GraphGibbs* zanemaril najdene vzorčne nize motiva $M_1$, saj najdena vzorčna množica ne bo zadostila danim pogojem.

V primeru iskanja več motivov hkrati je pri tej verziji algoritma *GraphGibbs* mogoče bolj smotrno prepustiti dolžine in število pojavitev samemu algoritmu, čeprav to lahko podceni dejansko dolžino motiva in število pričakovanih pojavitev. Drugače pa se je algoritem izkazal bolje, ko smo daljši motiv ali motiv z večjo vrednostjo $E$

določili kot prvi motiv v množici. Očiten vpliv vrstnega reda je potrebno dodatno obdelati in izboljšati algoritem, da se ne zanemari krajšega motiva (kot ga pričakujemo) z velikim številom pojavitev in s tem konča iskanje. To obravnavo pa prepustimo nadaljnjem razvijanju predlagane metode za iskanje motivov v DNA sekvencah.

### Realne množice

Prve tri predpostavke imajo še bolj izrazit vpliv pri realnih množicah podatkov, kjer so znani vzorčni nizi različnih dolžin in neenakomerno porazdeljeni. Prav tako smo obravnavali znane vzorčne nize kot pojavitve enega motiva, kar je lahko prispevalo k slabši občutljivosti predvsem na nukleotidnem nivoju. Prva predpostavka pa je prispevala k povečanemu številu lažno pozitivnih rezultatov, kar se je pokazalo pri nizki pozitivni napovedni vrednosti na sliki 5.20.




Slika 6.3: Primerjava uspešnosti algoritma *GraphGibbs* pri tretji nastavitvi prostih parametrov z vrednostmi statistik nSn, nPPV, nPC in nCC na nukleotidnem nivoju (stolpci) ter vrednosti statistik sSn, sPPV in sASP na vzorčnem nivoju (črte).

Prav tako smo primerjali različne kombinacije vrednosti parametrov $W$ in $E$ pri pogonu algoritma *GraphGibbs*, da smo zaznali vpliv teh dveh parametrov še posebej pri realnih množicah, kjer je dolžina znanih vzorčnih nizov ni enotna, kakor je tudi vprašljivo dejansko število pojavitev (enega izmed) motivov znotraj množice. Za primerjavo smo prikazali rezultate statistik pri posamezni kombinaciji algoritma na grafih C.10 ($W = 0$ & $E = 0$), 5.23 ($W = 0$ & $E = \sharp$), 6.3 ($W = \sharp$ & $E = 0$) in C.11 ($W = \sharp$ & $E = \sharp$). Za določitev dolžine motiva smo vzeli mediano dolžin vseh znanih vzorčnih nizov v množici. Najmanjša variabilnost v dolžinah vzorčnih nizov

pa je ravno pri sekvencah gliv, zato je pri slednjih množicah občutljivost algoritma *GraphGibbs* najvišja pri 3. in 4. kombinaciji na vzorčnem in nukleotidnem nivoju.

Primerjava med slikama C.10 (1.kombinacija) in 6.3 (3.kombinacija) poudari vpliv dolžine motiva na končno rešitev algoritma. Kadar je prevelika variabilnosti med dolžinami vzorčnih nizov, je bolje prepustiti algoritmu, da na podlagi modela grafa, ki zrcali notranjo strukturo danih sekvenc, določi najbolj primerno dolžino motiva. Potrebno pa je upoštevati, da je dobljena dolžina enotna za vse vzorčne nize, kar ima v posebnem vpliv na občutljivost na nukleotidnem nivoju.

Vrednost parametra $E$ ima največji vpliv pri množicah sekvenc muhe, pri katerih se je izkazalo, da je PPV algoritma *GraphGibbs* večja, kadar algoritem določi število pojavitev motiva glede na število pojavitev popolnoma ohranjenega vzorca, ki ga določimo v prvem delu. Pri ostalih skupinah množic je razlika med vrednostmi za PPV marginalna pri različnih kombinacijah. Nižjo pozitivno napovedno vrednost algoritma lahko pripišemo primarno naši prvi predpostavki pri razvoju algoritma, kot tudi dejstvu, da so lahko nekateri izmed najdenih vzorčnih nizov dejansko "pravi", saj so bili znani vzorčni nizi določeni ekperimentalno. Za izboljšavo PPV algoritma pa je potrebna obširnejša obdelava različnih realnih množic pri različnih nastavitvah algoritma, kot tudi ovržba oziroma omilitev predpostavke o vsaj eni pojavitvi motiva na sekvenco.

# 7. Zaključki

Za problem iskanja motivov v DNA sekvencah smo razvili metodo, pri kateri z modelom multigrafa analiziramo strukturo DNA sekvenc in uporabimo Gibbsovo vzorčenje za končno določitev poravnave iskanega motiva. Vplive sedmih prostih parametrov algoritma *GraphGibbs* na njegov rezultat smo raziskali z Plackett-Burmanovim eksperimentalnim načrtom. Najmočnejši vpliv na samo rešitev, ki smo jo ocenili s statistiko $I$ in odstotkom ujemanja $U$, sta imela dolžina motiva in število pričakovanih pojavitev motiva. Število iteracij pri Gibbsovem vzorčenju smo določili s pomočjo rezultatov Gelman-Rubinovega testa za preverjanje konvergence oziroma stacionarnosti Markovske verige, ki jo zgradi Gibbsov vzorčevalnik. Ostale parametre smo fiksirali na isto vrednost za testiranje uspešnosti algoritma na generiranih in realnih množicah.

S primerjalnimi grafi statistik za merjenje uspešnosti algoritma *GraphGibbs*, kot so občutljivost in pozitivna napovedna vrednost na vzorčnem nivoju ter koeficienta uspešnosti in korelacije na nukleotidnem nivoju, smo potrdili, da algoritem *GraphGibbs* zelo dobro določi popolnoma ohranjene motive. Prav tako je uspešen pri relativno ohranjenih motivih, ki vsebujejo več kot devet črk in so enakomerno porazdeljeni v množici približno desetih sekvenc. Največja razpršenost rezultatov navedenih statistik je bila pričakovano med množicami z najkrajšo dolžino motiva ($W = 6$). Kratka dolžina motiva namreč upočasni razvoj dveh dinamičnih struktur, to sta pozicijsko utežena matrika $\mathcal{Q}$ in poravnava motiva, ki ju vodi Gibbsov vzorčevalnik.

Z večanjem dolžine znanega motiva se izboljša uspešnost algoritma *GraphGibbs* kot tudi zmanjša razpršenost rezultatov. Najboljši rezultati pri generiranih množicah z enim motivom so bili pri množicah z najdaljšim motivom ($W = 18$); tudi pri višjih stopnjah variabilnosti. Prvi del algoritma tako uspešno zazna krajši popolnoma ohranjen podniz znotraj daljših znanih vzorčnih nizov, katerega poravnava je dobra začetna točka za Gibbsov vzorčevalnik v drugem delu algoritma *GraphGibbs*, kjer najdemo še vse variacije motiva.

Analiza rezultatov pri generiranih množicah z dvema motivoma je dodatno izpostavila stopnjo variabilnosti in zaporedje iskanja motivov, ki poleg dolžine motiva in števila njegovih pričakovanih pojavitev znotraj množice vplivajo na izhod algoritma *GraphGibbs*. Iskanje je bolj uspešno, kadar se najprej išče daljši motiv oziroma motiv z večjim številom pričakovanih pojavitev. Druga možnost je, da podcenimo pravo

dolžino ter s tem omilimo pogoje določanja vzorčne množice in dobimo kandidata za motiv kljub slabši občutljivosti na nukleotidnem nivoju.

Obstaja še možnost, da prepustimo algoritmu *GraphGibbs*, da sam določi primerno dolžino za motiv. Takšna nastavitev je tudi najbolj primerna za obdelavo realnih podatkov, kjer dolžina vzorčnih nizov ni enotna. Čeprav algoritem ponavadi podceni pravo dolžino, je pri heterogenih množicah bolje privzeti dolžino motiva, ki jo dobimo po analizi zaporedja nukleotidov v DNA sekvencah v prvem delu algoritma. V teh primerih algoritem zazna večji delež znanih vzorčnih nizov, vendar jih ne zazna v celoti. Izmed množic sekvenc človeka, miši, muhe in glive je bil algoritem v povprečju najbolj uspešen na množicah muhe. Če smo dolžino motiva določili, pa je algoritem imel največjo občutljivost na množicah sekvenc glive.

Pri vseh štirih možnih nastavitvah algoritma *GraphGibbs* pri obdelavi realnih podatkov je v povprečju pozitivna napovedna vrednost algoritma opazno nižja od občutljivosti algoritma na obeh nivojih. Torej je algoritem pri iskanju motiva in njegovih pojavitev v realnih DNA sekvencah določil večjo vzorčno množico kot je znana, saj je bilo več lažno pozitivnih zadetkov. To je posledica tako nastavitev algoritma, pogojev iskanja, kot tudi določanja znanih pojavitev funkcionalnega motiva v segmentih DNA sekvenc z eksperimentalnim postopkom.

Pogoji iskanja motiva so v prvem delu določeni z osnovnimi predpostavkami, ki smo jih privzeli pri razvijanju metode. V toku samega raziskovanja smo se srečali tudi z omejenostjo podatkovnih in literarnih virov na temo iskanja motivov v DNA sekvencah, kar je vplivalo na naše modeliranje zelo kompleksnega problema. Nekaj pomembnejših problemov pri razvoju metode je izpostavljenih v naslednjem razdelku. Doktorsko delo pa zaključimo z idejami za nadaljnji razvoj naše metode za iskanje neznanih motivov v DNA sekvencah, s katerimi lahko izboljšamo uspešnost algoritma *GraphGibbs*.

## 7.1 Omejitve raziskave in problemi

Metodo za iskanje motivov transkripcijskih dejavnikov v DNA sekvencah smo raziskovali predvsem z matematičnega oziroma statističnega vidika. Interpretacija vmesnih rezultatov pri razvijanju algoritma in končnih rezultatov za preverjanje uspešnosti algoritma na različnih množicah pa zahteva tudi poglobljeno znanje genetike in pravil uravnavanja izražanja genov. Interpretacija je velik del razvijanja tovrstnih orodij za iskanje vzorcev, saj imajo lahko podatki velikokrat določene vzorce, ki nimajo želene funkcije oziroma niso del raziskave. Ločevanje med "zanimivimi" (dobrimi) vzorci in "nezanimivimi" (slabimi) vzorci je velikokrat samostojen problem pri zaznavanju vzorcev tako pri molekularni genetiki kot pri ostalih raziskovalnih smereh. Zato smo se v tem delu odločili za preverjanje algoritma na generiranih podatkovnih množicah, ki predstavljajo dobro kontrolno testno okolje, kot tudi na že testiranih realnih množicah podatkov z določenimi motivi.

Generiranje lastnih podatkov nam je omogočilo večjo kontrolo pri testiranju vpliva

različnih lastnostih podatkovnih množic, kot na primer število sekvenc v množici, dolžina sekvenc, število pojavitev motiva in njihova porazdelitev po sekvencah, dolžino vzorčnih nizov in stopnjo variabilnosti le teh v celotni množici. S kontrolirano spremembo teh lastnosti smo lahko simulirali različne lastnosti realnih DNA sekvenc, ki so nam bile na voljo. Kljub temu, da simulirani podatki podajo veliko informacij o uspešnosti in delovanju našega algoritma, pa je en izmed naših ciljev postaviti model metode, ki je uporaben na realnih podatkih.

Raziskovalci imajo sedaj dostop do velikih genskih baz, kamor se dosledno vključujejo nove DNA, RNA in proteinske sekvence različnih oraganizmov in dopolnjujejo starejši podatki. Takšna baza je tudi TRANSFAC genska baza, kjer so zbrani deli DNA sekvenc s promotorji transkripcijskih dejavnikov. To bazo sestavljajo sekvence evkariontskih organizmov, primarno sesalcev in rastlin. Pojavitve vzorčnih nizov motiva, ki nas zanima, so določene eksperimentalno, s katerimi se določi presečni vzorec oziroma motiv. Vzorčni nizi in motiv so nato podlaga za iskanje novih, še ne zaznanih, vzorčnih nizov istega motiva, kot tudi odkrivanje funkcij teh zapisov pri uravnavanju genskega izražanja. Iz te baze je Tompa s sodelavci [48], sestavil testne množice podatkov za primerjalno analizo uspešnosti delovanja algoritmov za iskanje motivov v DNA sekvencah.

Tompa s sodelavci je v svojem članku [48] med prvimi objavil rezultate statistične analize uspešnosti posameznih iskalnih algoritmov na istih testnih množicah. Njihova testna baza je prosto dostopna in namenjena uporabi analize delovanja novih metod za iskanje motivov. V ta namen smo jo uporabili tudi pri našem raziskovanju. Interpretacija in ekstrakcija primernih sekvenc iz genskih baz, kot je na primer TRANSFAC, za gradnjo nove samostojne testne baze namreč zahteva veliko poznavanja genskih sekvenc različnih vrst organizmov kot tudi poglobljeno poznavanje posameznih podatkovnih baz. Raziskovanje genskih baz in realnih sekvenc je lahko samostojen raziskovalni projekt, ki pa presega časovne in znanstvene okvirje tega doktorskega dela. Predvsem zaradi časovne omejitve našega projekta in pomanjkanja potrebnega znanja za gradnjo samostojnih baz, smo se odločili za uporabo že sestavljene baze realnih množic.

Ker baze realnih množic nismo zgradili sami, moramo privzeti, da so dani vzorčni nizi pri posamezni množici pravilno določeni. Pojavitve vzorčnih nizov so določene sicer eksprerimentalno, vendar moramo dopustiti možnost, da kakšen vzorčni niz ni bil zaznan, ali da je bil napačno določen, kar je lahko med drugim posledica merskih ali sistematskih napak. Podatki v genski bazi TRANSFAC imajo tudi precej dolge dolžine vzorčnih nizov. Tako imajo realne množice, ki smo jih predstavili v tabeli 4.3 v tem delu, do 35 vzorčnih nizov dolžine $31 - 71$. Tako dolge dolžine imajo lahko negativen vpliv pri merjenju občutljivosti algoritma.

Glavna omejitev pri testiranju algoritma nam je predstavljalo pomanjkanje realnih testnih množic različnih organizmov. Dodatno je pomanjkanje podatkovnih baz, kot so jo naredili Tompa in sodelavci, ki bi služile testiranju novih metod s primerjavo rezultatov že uveljavljenih metod. Tompa je s sodelavci začel postavljati grob regulatorni okvir za merjenje uspešnosti novega algoritma, vendar je to področje močno

pomanjkljivo in še nerazvito. Čeprav se količina podatkov realnih sekvenc raznih organizmov veča in so ti bolj dostopni, je metodologija za njihovo obdelavo še vedno relativno mlada. V prvem poglavju smo predstavili nekaj objavljenih in uporabnih metod za iskanje motivov, vendar je potrebna izboljšava teh metod; med drugim optimizacija časovne in prostorske zahtevnosti ter izboljšava natančnosti algoritmov na različnih vrstah organizmov, tako nižjih (prokarionti) kot višjih vrstah (evkarionti).

Raziskave na molekularni genetiki so sedaj v razcvetu. Veliko raziskovalnih problemov s tega področja spada med tako imenovane probleme velikih podatkov (ang. big data problems), kjer so na voljo ogromne količine podatkov, ampak ne poznamo še vseh njihovih lastnosti in funkcij kot tudi narave povezav med njimi. To pomanjkanje znanja otežuje modeliranje procesov in interpretacijo rezultatov. Kljub velikemu raziskovalnemu napredku na tem področju, obstaja raziskovanje DNA sekvenc in regulatornih sistemov celice relativno kratko obdobje, kar pomeni, da še niso uspeli zgraditi referenčnih podatkovnih baz ali določiti vseh pomembnih področij v posameznih genomih. S tehničnim napredkom pri sekvencioniranju se je začela razvijati filogenija in primerjalna genetika, kar ponuja nove vidike pri raziskovanju uravnavanja genskega izražanja, genske evolucije vrst in povezavo med posameznimi organizmi na molekularni ravni celice. Te raziskave pa so še v povojih, zato je tudi literatura temu primerno razpršena in pomanjkljiva. Slovenske literature na to problematiko pa je izredno malo, kar je predstavljalo manjšo omejitev predvsem v začetni fazi raziskovalnega dela.

## 7.2 Izboljšave algoritma

Kot smo že videli pri Plackett-Burmanovem eksperimentalnem načrtu ima predvidena dolžina motiva vpliv na rezultat algoritma *GraphGibbs*. Predpostavka o enotni dolžini algoritma olajša modeliranje sekvenc, vendar postavi dodatne omejitve pri iskanju motivov. Za določanje optimalne dolžine motiva $W$ priporočamo tako uporabo dolžine popolnoma ohranjenega vzorca, ki ga dobimo v prvem delu algoritma, kot tudi primerjavo različnih vrednosti $W$ na podlagi statistike $I$ v drugem delu algoritma, s tem da upoštevamo poravnavo popolnoma ohranjenega vzorca. Pri takšni kombinaciji bi uporabili dolžino $\ell_v$ in poravnavo popolnoma ohranjenega vzorca $v$ kot vodilo pri iskanju vrednosti $W \geq \ell_v$, ki maksimizira statistiko $I$. Možne dolžine določi uporabnik sam ali pa privzamemo interval $[\ell_v, 30]$.

Prvi del algoritma *GraphGibbs* določa potek iskanja in pri kratkih dolžinah motiva pogosto konča iskanje, kar smo videli pri krožnih diagramih; predvsem ko so motivi popolnoma ohranjeni. Vzrok je v nastavitvi mej za zaključek iskanja. Spodnja meja je določena z $\lfloor E/2 \rfloor$, kjer je $E$ pričakovano število pojavitev motiva, ki je pri osnovni nastavitvi enako številu sekvenc v množici $N$, saj smo predpostavili, da ima vsaka sekvenca vsaj eno pojavitev. Zgornja meja je nedefinirana, ker želimo najti čim večje število pojavitev najdaljšega (popolnoma ohranjenega) motiva.

Kadar je pričakovano število $E$ veliko, potem je spodnja meja za število pojavi-

tev v prvem delu visoka. V prvem delu algoritem išče popolnoma ohranjene vzorce, vendar se le-ti redkeje pojavijo v množici. Prav tako je število pojavitev popolnoma ohranjenega vzorca v obratnem razmerju z dolžino vzorca. V primeru, ko je predvidena prevelika dolžina motiva, algoritem ne najde vzorca, ki ima zadostno število pojavitev; zato iskanje konča. To smo lahko zaznali tudi pri množicah z dvema motivoma. Odvisnost med parametroma $E$ in $W$ je potrebno podrobneje analizirati ter vključiti v naš model, tako da algoritem ne končna iskanja predčasno.

Prav tako moramo omiliti predpostavko, da vsaka od sekvenc vsebuje vsaj eno pojavitev motiva, saj pri neenakomerno porazdeljenih motivih preveč vpliva na izhod algoritma in algoritem določi več lažno pozitivnih zadetkov, kar pa vliva na njegovo pozitivno napovedno vrednost. To lahko omogočimo z modifikacijo porazdelitvene funkcije $\Gamma$, tako da upoštevamo $P(E_i = 0) > 0$ za vsako od sekvenc $i$. S tem bi dopustili tudi možnost, da v sekvenci ni nobene pojavitve motiva.

Kadar z Gibbsovim vzorčevalnikom nismo zaznali znane pojavitve motiva, je lahko algoritem deloma zaznal motiv s poravnavo prvega dela. Slednjo je vredno podrobneje pogledati, saj je zaradi osnovnih predpostavk to poravnava vzorca z nadpovprečnim številom pojavitev. Poleg vrednotenja poravnave med samim Gibbsovim vzorčenjem, manjka metodi kriterij za rangiranje najdenih motivov od najbolj verjetnega do najmanj verjetnega za iskano rešitev. Z integracijo takšnega kriterija bi prepustili, da algoritem določi vse možne motive in v primeru, da je uporabnik določil število znanih motivov, podamo za rešitev poravnavo motiva z najboljšo oceno.

Med drugim predlagamo še obravnavo bolj robustne statistike za ocenjevanje poravnave namesto statistike $I$. Plackett-Burmanov načrt in rezultati algoritma *GraphGibbs* so izpostavili občutljivost statistike $I$ na dolžino motiva $W$ in število najdenih vzorčnih nizov $E$. Primernost statistike $I$ in nekaterih ostalih statistik za ocenjevanje poravnave so že statistično obdelali s podatkovnimi množicami, ki smo jih uporabili tudi pri našem raziskovanju. Rezultati obdelave so objavljeni v članku [30]. Podobne obdelave bi morali nadaljevati še na drugih podatkovnih množicah, da ne pride do prevelikega prilagajanja modela k podatkom. Raznolikost podatkovnih množic realnih podatkov je pomembna tako pri samem modeliranju kot pri optimizaciji natančnosti in hitrosti metode. Samo zbiranje teh množic in njihova obdelava pa sta že samostojna raziskovalna podproblema pri kompleksnem problemu iskanja motivov v DNA sekvencah.

# Literatura in Viri


[1] D.J. Allocco, I.S. Kohane in A.J. Butte. "Quantifying the relationship between co-expression, co-regulation and gene function". V: *BMC Bioinformatics* (2004).

[2] T.L. Bailey in C. Elkan. "Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization". V: *Machine Learning* 21 (1995), str. 50–83.

[3] C.K. Bayne in I.B. Rubin. *Practical experimental designs and optimization methods for chemists*. Prva. Florida, USA: VCH Publishers, Inc., 1986.

[4] C.M. Bishop. *Pattern Recognition and Machine Learning*. Prva. New York, USA: Springer Science+Business Media, LLC, 2006.

[5] W.M. Bolstad. *Introduction to Bayesian Statistics*. Druga. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 2007.

[6] X. Chen in T. Jiang. "An improved Gibbs sampling method for motif discovery via sequences weighting". V: *Computational Systems Bioinformatics*. Ur. P. Markstein in Y. Xu. Stanford, CA: Imperial College Press, 2006, str. 239–247.

[7] T.H. Cormen in sod. *Introduction to Algorithms*. Druga. Cambridge, England: The MIT Press, 2001.

[8] F.H. Crick, L. Barnett, S. Brenner in sod. "General nature of the genetic code for proteins". V: *Nature* 4809 (1961), str. 1227–1232.

[9] M.K. Das in H. Dai. "A survey of DNA motif finding algorithms". V: *BMC Bioinformatics* 8 (2007). suppl. 7.

[10] M. Defrance in J. van Helden. "info-gibbs: a motif discovery algortihm that directly optimized information content during sampling". V: *Bioinformatics* 25 (2009), str. 2715–2722.

[11] S.N Deming in S.L. Morgan. *Experimental design: a chemomentric approach*. Druga. Amsterdam, The Netherlands: Elsevier Science Publishers B.V., 1993.

[12] M. Dermastia, R. Komel in T. Turk. *Kjer se življenje začne*. Prva. Ljubljana, Slovenija: Rokus Klett, 2011.

[13] J.L. Devore. *Probability and Statistics for Engineering and the Sciences*. Četrta. USA: Brooks/Cole Publishing Company, 1995.

[14] A. Gelman in sod. *Bayesian Data Analysis*. Druga. Boca Raton, Florida, USA: Chapman & Hall/CRC, 2004.

[15] M. Gupta in J.S. Liu. *Bayesian inference for gene expression and proteomics.* Cambridge: Cambridge University Press, 2006.

[16] J. van Helden, B. André in J. Collado-Vides. "Extracting Regulatory Sites from the Upstream Region of Yeast Genes by Computational Analysis of Oligonucleotide Frequencies". V: *J. Mol. Biol.* 281 (1998), str. 827–842.

[17] G.Z. Hertz, G.W. Hartzell in G.D. Stormo. "Identification of consensus patterns in unaligned DNA sequences known to be functionally related". V: *Comput. Appl. Biosci.* 6 (1990), str. 81–92.

[18] G.Z. Hertz in G.D. Stormo. "Identifying DNA and protein patterns with statistically significant alignments of multiple sequences". V: *Bioinformatics* 15 (1999), str. 563–577.

[19] Y.V. Heyden in sod. *Guidance for Robustness/Ruggedness Test in Method Validation.* Prva. Vrije Universiteit Brussel in Unilever Research Vlaardingen. Mar. 2006.

[20] R. Jamnik. *Matematična statistika.* Prva. Ljubljana, Slovenija: Državna založba Slovenije, 1980.

[21] R. Komel. *GENETIKA od vijačnice do kloniranja.* Prva. Ljubljana, Slovenija: Založba Rokus, 2006.

[22] M. Kon, D. Holloway in C. DeLisi. "SVMMotif: A machine learning motif algorithm". V: *International Conference on Machine Learning and Application* 6 (2007), str. 573–580.

[23] B. Košmelj in sod. *Statistični terminološki slovar.* Razširjena. Ljubljana, Slovenija: Statistično društvo Slovenije in Statistični urad Republike Slovenije, 2001.

[24] J. Kozak. *Podatkovne strukture in algoritmi.* Prva. Ljubljana, Slovenija: Društvo matematikov, fizikov in astronomov, 1986.

[25] D.P. Kroese, Taimre T. in Botev Z.I. *Handbook of Monte Carlo Methods.* Prva. New Jersey, USA: John Wiley&Sons,Inc., 2011.

[26] D.S. Latchman. *Gene control.* Prva. USA: Garland Science, 2010.

[27] C.E. Lawrence, M.S. Altschul in sod. "Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment". V: *Science* 8 (1993), str. 208–214.

[28] C.E. Lawrence in J.S. Liu. "Bayesian inference on biopolymer models". V: *Bioinformatics* 15 (1999), str. 38–52.

[29] C.E. Lawrence in A.A. Reilly. "An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences". V: *Proteins* 7 (1990), str. 41–51.

[30] Nan Li in M. Tompa. "Analysis of computational approaches for motif discovery". V: *Algorithms for Molecular Biology* 1 (2006).

[31] D. Liu, X. Xiong, B. DasGupta in sod. "Motif Discoveries in Unaligned Molecular Sequences Using Self-Organizing Neural Networks". V: *IEEE Trans. Neural Networks* 17 (2006), str. 919–928.

[32] J.S. Liu, M. Gupta, X. Liu in sod. "Statistical models for biological sequence motif discovery". V: *Case Studies in Bayesian Statistics*. Ur. C. Gatsonis, R.E. Kass, A. Carriquiry in sod. Pittsburg, PA: Carnegie Mellon University, 2002, str. 1–18.

[33] J.S. Liu in T. Logvinenko. "Appendix A: Markov Chain Monte Carlo Methods". V: *Handbook of Statistical Genetics*. Ur. D.J. Balding, M. Bishop in C. Cannings. Tretja. Hoboken: John Wiley & Sons Ltd, 2007, str. 91–92.

[34] X. Liu, D.L. Brutlag in J.S. Liu. "Bioprospector: Discovering Conserved DNA Motifs in Upstream Regulatory Regions of Co-expressed Genes". V: *Pac. Symp. Biocomput.* 6 (2001), str. 127–138.

[35] P. Lukšič. "Rast v grafih". Doktorska disertacija. Ljubljana, Slovenija: Fakulteta za računalništvo in informatiko, 2009.

[36] M. Žakelj Mavrič in M. Dolinar, ur. *ANGLEŠKO-slovenski slovar izbranih izrazov iz biokemije in molekularne biologije*. Slovensko biokemijsko društvo, 2012.

[37] R. Požar. "Kompleksnost in algoritminčni problemi v teoriji krovnih grafov". Doktorska disertacija. Ljubljana, Slovenija: Fakulteta za računalništvo in informatiko, 2013.

[38] N. Prijatelj. *Matematične strukture I*. Druga. Ljubljana, Slovenija: Mladinska knjiga, 1971.

[39] P. Resnik in E. Hardisty. *Gibbs Sampling for the Uninitiated*. Teh. poročilo LAMP-TR-153. University of Maryland, College Park: Department of Linguistics, Institute for Advanced Computer Studies, 2010.

[40] F.P. Roth in sod. "Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation". V: *Nature Biotechology* 16 (1998), str. 939–945.

[41] K. Shida. "GibbsST: a Gibbs sampling method for motif discovery with enhanced resistance to local optima". V: *BMC Bioinformatics* 7 (2006), str. 486.

[42] A.P. Smith. "Nucleic Acids to Amino Acids: DNA Specifies Protein". V: *Nature Education* 1 (2008), str. 126.

[43] Y. Song, C. James in A.Y. Chi. "A new approach for identifying transcription factor binding sites". V: *International Journal of Science & Informatics* 2 (2012), str. 15–22.

[44] Ž. Stepančič. "Enhancing Gibbs Sampling Method for Motif Finding in DNA with Initial Graph Representation of Sequences". V: *Journal of Computational Biology* 21 (2014), str. 1–12.

[45] G. Stormo. "DNA binding sites: representation and discovery". V: *Bioinformatics* 16 (2000), str. 16–32.

[46] P. Stušek in B. Vilhar. *Biologija celice in genetika*. Prva. Ljubljana, Slovenija: DZS, 2010.

[47] G. Thijs, K. Marchal, M. Lescot in sod. "A Gibbs Sampling Method to Detect Overrepresented Motifs in the Upstream Regions of Coexpressed Genes". V: *Journal of Computational Biology* 9 (2002), str. 447–464.

[48] M. Tompa, N. Li, T.L. Bailey in sod. "Assessing computational tools for the discovery of transcriptions factor binding sites". V: *Nature Biotehnology* 23 (2005), str. 137–144.

[49] Ministerstvo za šolstvo in šport. *Filogenija*. 2015. URL: http://mss.svarog.si/biologija/MSS/index.php?page_id=11534.

[50] Ministerstvo za šolstvo in šport. *Klasična genetika*. 2015. URL: http://mss.svarog.si/biologija/MSS/index.php?page_id=11283.

[51] J.D. Watson. *DNK - Skrivnost Življenja (prevod)*. Prva. Ljubljana, Slovenija: Modrijan založba, 2007.

[52] J.D. Watson in F.H.D. Crick. "Molecular structure of nucleic acids". V: *Nature* 4356 (1963).

[53] D.B. West. *Introduction to Graph Theory*. Druga. Delhi, India: Pearson Education, Inc., 2001.

[54] Wikipedia. *DNA*. 2015. URL: http://en.wikipedia.org/wiki/DNA.

[55] J. Zupan. *Kemometrija in obdelava eksperimentalnih podatkov*. Prva. Zavod za humanistiko in Kemijski inštitut Ljubljana, Ljubljana: Inštitut Nove Revije, 2009.

# Priloge

# A. Izpis algoritma *GraphGibbs*

K primeru 3.2.1 smo dodali izpis algoritma *GraphGibbs*, kjer je prikazana izvedba postopka na konkretnem zgledu. Poudarek je na prvem delu algoritma *GraphGibbs*.

Dano imamo množico petih sekvenc. Vsaka od sekvenc vsebuje 109 nukleotidov. Motiv, ki nas zanima, ima zaporedje nukleotidov GCTATTCGG in je popolnoma ohranjen. V tem primeru se pojavi v vsaki sekvenci samo enkrat, tako da je pričakovano število pojavitev $E = 5$.

Prvi del se začne s kodiranjem sekvenc in gradnje matrike povezav. Glavne informacije v matriki povezav so podane v obliki tabele s stolpci `vrsta` (vrstica matrike), `# pov` (največje število v vrstici matrike povezav), `max_vred` (največja normalizirana vrednost v vrstici) in `sosedi` (številka stolpca, ki vsebuje največjo normalizirano vrednost). Z zeleno barvo je označenih prvih deset vrstic z največjimi normaliziranimi vrednostmi.

Izbrana vozlišča predstavljajo začetna vozlišča za usmerjene sprehode, ki jih določimo s prilagojeno metodo *Iskanje v širino*. Algoritem *GraphGibbs* nato nadaljuje s postopkom iskanja kandidata za končen motiv. Za vsako množico $\mathcal{K}_\delta$, to je množica vozlišč s številom obiskov $\delta$, so izpisani možni kandidati. Vsak kandidat je podan v kodirani in navadni obliki. Na podlagi kandidatov začne algoritem graditi motiv do izbrane dolžine; v tem primeru je to $W = 9$. Celoten postopek v šestih korakih je opisan v podpoglavju 3.2.

Ker smo iskali popolnoma ohranjen motiv, so bile vse pojavitve motiva določene v prvem delu algoritma. Kot rešitev poda algoritem poravnavo motiva (vse vzorčne nize in njihove položaje v sekvencah). Dodatno so določene še pozicijsko utežena matrika $\mathcal{Q}$ in frekvence ozadja pri končni poravnavi ter presečni vzorec, ki predstavlja rešitev.

Vhodna datoteka: ./set/T1_9_0_U_1_5_1c.txt
Algoritem: GraphGibbs
Verzija: 6

Datum: 14/1/2015
Čas: 2:55:17

---

Vhodni parametri:
1) Število motivov: 1
2) Dolžina motiva: 9
3) Število pričakovanih pojavitev: 1
4) Porazdelitev pojavitev: u
5) Število iteracij: 100
6) R/F: F
7) B: sqrt{6}

---

** Obnova vhodnih podatkov. **

Število sekvenc: 5
Število črk: 545
Dolžine sekvenc:
sek1: 109
sek2: 109
sek3: 109
sek4: 109
sek5: 109

Frekvence nukleotidov in psevdoštevila.

B: 2.23607
Vseh črk je: 545

| črka | frekvenca | psevdoštevilo |
|---|---|---|
| A | 0.245872 | 0.549786 |
| T | 0.233028 | 0.521065 |
| C | 0.240367 | 0.537477 |
| G | 0.280734 | 0.62774 |

* PRVI DEL *

---

MOTIV 1

Število možnih motivov: 1
Število pričakovanih pojavitev: 5
Porazdelitev števila pričakovanih pojavitev: u

Iz matrike povezav:

| vrsta | #pov | max_vred | sosedi |
|---|---|---|---|
| 1 | 1 | 0.142857 | 1 3 5 10 18 39 58 |
| 2 | 1 | 0.142857 | 8 12 29 30 43 44 52 |
| 3 | 1 | 0.0833333 | 3 12 16 27 30 32 34 36 53 56 60 61 |
| 4 | 1 | 0.25 | 30 37 50 64 |
| 5 | 1 | 0.2 | 6 29 45 56 60 |
| **6** | **5** | **0.454545** | **48** |
| 7 | 2 | 0.153846 | 44 |
| 8 | 2 | 0.2 | 56 |
| **9** | **3** | **0.3** | **61** |
| 10 | 2 | 0.285714 | 61 |
| 11 | 1 | 0.142857 | 8 14 15 39 48 49 53 |
| 12 | 1 | 0.142857 | 8 14 17 32 46 50 52 |
| 13 | 1 | 0.142857 | 12 14 17 31 35 36 62 |
| 14 | 2 | 0.222222 | 53 |
| 15 | 2 | 0.222222 | 18 |
| 16 | 2 | 0.25 | 7 |
| 17 | 1 | 0.25 | 24 29 40 46 |
| **18** | **5** | **0.454545** | **28** |
| 19 | 1 | 0.2 | 13 16 45 46 52 |
| 20 | 2 | 0.2 | 9 48 |
| **21** | **1** | **0.333333** | **8 29 60** |
| 22 | 1 | 0.25 | 28 48 53 56 |
| 23 | 2 | 0.2 | 61 63 64 |
| 24 | 1 | 0.166667 | 11 19 29 32 48 55 |
| 25 | 2 | 0.25 | 28 |
| 26 | 1 | 0.25 | 8 37 50 59 |
| 27 | 2 | 0.181818 | 42 47 49 |
| 28 | 2 | 0.153846 | 58 64 |
| 29 | 2 | 0.181818 | 25 |
| 30 | 1 | 0.1 | 3 9 11 13 16 49 50 53 57 61 |
| 31 | 1 | 0.166667 | 8 15 18 26 46 62 |
| 32 | 2 | 0.2 | 29 |
| 33 | 1 | 0.2 | 24 27 36 41 47 |
| 34 | 1 | 0.125 | 14 31 34 43 46 48 50 56 |
| 35 | 1 | 0.2 | 22 30 42 44 61 |
| 36 | 1 | 0.1 | 5 18 30 37 41 42 44 49 50 51 |
| **37** | **5** | **0.5** | **23** |
| **38** | **1** | **0.5** | **15 30** |
| 39 | 2 | 0.25 | 45 |
| 40 | 1 | 0.142857 | 3 7 20 30 44 49 52 |
| **41** | **2** | **0.285714** | **59** |
| 42 | 1 | 0.166667 | 14 20 34 36 44 49 |
| 43 | 1 | 0.1 | 15 20 25 27 37 39 57 59 62 63 |
| 44 | 2 | 0.166667 | 3 |
| 45 | 2 | 0.166667 | 16 |
| 46 | 1 | 0.1 | 3 7 11 16 17 18 35 45 61 64 |
| 47 | 1 | 0.25 | 6 44 50 53 |
| 48 | 2 | 0.133333 | 37 63 |
| 49 | 1 | 0.0714286 | 1 2 15 19 27 33 35 40 41 46 48 56 61 64 |

```
50    1    0.0769231  2  15 22 26 27 33 34 43 45 48 57 58 62
51    2    0.222222   10
52    1    0.111111   3  8  16 19 24 27 30 46 53
53    2    0.153846   36 44
54    1    0.166667   2  8  30 33 34 62
55    1    0.2        11 33 47 51 52
56    2    0.142857   52
57    1    0.333333   24 30 58
58    5    0.5        6
59    1    0.166667   7  21 29 48 49 54
60    1    0.111111   2  4  5  20 21 25 49 53 59
61    1    0.0666667  4  7  9  11 23 27 28 32 35 39 49 52 53 54
                 60
62    2    0.222222   25
63    3    0.333333   20
64    2    0.25       60


Prvih 10 vozlišč:

58 37 38 6 18 21 63 57 9 41


        1) Dolžina motiva: 9

Število trojčkov: 3
Število pričakovanih pojavitev: 5

Usmerjen sprehodi:

#    sprehod         sekvenca
-----------------------------
1    58-> 6->48      GCTATTCGG
2    37->23->61      CTATTCGGA
3    37->23->63      CTATTCGGC
4    37->23->64      CTATTCGGG
5    38->15->18      CTTAGCTAT
6    38->30->3       CTTTGTAAC
7    38->30->9       CTTTGTACA
8    38->30->11      CTTTGTACC
9    38->30->13      CTTTGTAGA
10   38->30->16      CTTTGTAGG
11   38->30->49      CTTTGTGAA
12   38->30->50      CTTTGTGAT
13   38->30->53      CTTTGTGTA
14   38->30->57      CTTTGTGCA
15   38->30->61      CTTTGTGGA
16    6->48->37      ATTCGGCTA
17    6->48->63      ATTCGGGGC
18   18->28->58      TATTCGGCT
19   18->28->64      TATTCGGGG
20   21-> 8->56      TTAATGGTG
```

| | | |
|---|---|---|
| 21 | 21->29->25 | TTATGATCA |
| 22 | 21->60->2 | TTAGCGAAT |
| 23 | 21->60->4 | TTAGCGAAG |
| 24 | 21->60->5 | TTAGCGATA |
| 25 | 21->60->20 | TTAGCGTAG |
| 26 | 21->60->21 | TTAGCGTTA |
| 27 | 21->60->25 | TTAGCGTCA |
| 28 | 21->60->49 | TTAGCGGAA |
| 29 | 21->60->53 | TTAGCGGTA |
| 30 | 21->60->59 | TTAGCGGCC |
| 31 | 63->20->9 | GGCTAGACA |
| 32 | 63->20->48 | GGCTAGCGG |
| 33 | 57->24->11 | GCATTGACC |
| 34 | 57->30->3 | GCATGTAAC |
| 35 | 57->58->6 | GCAGCTATT |
| 36 | 57->24->19 | GCATTGTAC |
| 37 | 57->24->29 | GCATTGTGA |
| 38 | 57->24->32 | GCATTGTGG |
| 39 | 57->24->48 | GCATTGCGG |
| 40 | 57->24->55 | GCATTGGTC |
| 41 | 57->30->9 | GCATGTACA |
| 42 | 57->30->11 | GCATGTACC |
| 43 | 57->30->13 | GCATGTAGA |
| 44 | 57->30->16 | GCATGTAGG |
| 45 | 57->30->49 | GCATGTGAA |
| 46 | 57->30->50 | GCATGTGAT |
| 47 | 57->30->53 | GCATGTGTA |
| 48 | 57->30->57 | GCATGTGCA |
| 49 | 57->30->61 | GCATGTGGA |
| 50 | 9->61->4 | ACAGGAAAG |
| 51 | 9->61->7 | ACAGGAATC |
| 52 | 9->61->9 | ACAGGAACA |
| 53 | 9->61->11 | ACAGGAACC |
| 54 | 9->61->23 | ACAGGATTC |
| 55 | 9->61->27 | ACAGGATCC |
| 56 | 9->61->28 | ACAGGATCG |
| 57 | 9->61->32 | ACAGGATGG |
| 58 | 9->61->35 | ACAGGACAC |
| 59 | 9->61->39 | ACAGGACTC |
| 60 | 9->61->49 | ACAGGAGAA |
| 61 | 9->61->52 | ACAGGAGAG |
| 62 | 9->61->53 | ACAGGAGTA |
| 63 | 9->61->54 | ACAGGAGTT |
| 64 | 9->61->60 | ACAGGAGCG |
| 65 | 41->59->7 | CCAGCCATC |
| 66 | 41->59->21 | CCAGCCTTA |
| 67 | 41->59->29 | CCAGCCTGA |
| 68 | 41->59->48 | CCAGCCCGG |
| 69 | 41->59->49 | CCAGCCGAA |
| 70 | 41->59->54 | CCAGCCGTT |

Največje število obiskov: 5

| # | v_i | kandidati | trojčki |
|---|---|---|---|
| 1 | 9 | 9 -> 61 | ACAGGA |
| 2 | 48 | 48 -> 37 | CGGCTA |
| 3 | | 48 -> 63 | CGGGGC |
| 4 | 49 | 49 -> 1 | GAAAAA |
| 5 | | 49 -> 2 | GAAAAT |
| 6 | | 49 -> 15 | GAAAGC |
| 7 | | 49 -> 19 | GAATAC |
| 8 | | 49 -> 27 | GAATCC |
| 9 | | 49 -> 33 | GAACAA |
| 10 | | 49 -> 35 | GAACAC |
| 11 | | 49 -> 40 | GAACTG |
| 12 | | 49 -> 41 | GAACCA |
| 13 | | 49 -> 46 | GAACGT |
| 14 | | 49 -> 48 | GAACGG |
| 15 | | 49 -> 56 | GAAGTG |
| 16 | | 49 -> 61 | GAAGGA |
| 17 | | 49 -> 64 | GAAGGG |

Kandidati za motiv.

| C | # | motiv | W | našli | našli % |
|---|---|---|---|---|---|
| 1 | 1 | ACAGGA | 6 | 3/5 | 60 |
| | 2 | AACAGGA | 7 | 1/5 | 20 |
| | 3 | TACAGGA | 7 | 1/5 | 20 |
| | 5 | GACAGGA | 7 | 1/5 | 20 |
| | 6 | ACAGGAA | 7 | 1/5 | 20 |
| | 7 | ACAGGAT | 7 | 1/5 | 20 |
| | 8 | ACAGGAC | 7 | 1/5 | 20 |
| 2 | 1 | CGGCTA | 6 | 2/5 | 40 |
| | 3 | TCGGCTA | 7 | 2/5 | 40 |
| | 5 | TTCGGCTA | 8 | 2/5 | 40 |
| | 6 | ATTCGGCTA | 9 | 1/5 | 20 |
| | 7 | TTTCGGCTA | 9 | 1/5 | 20 |
| | 11 | TTCGGCTAT | 9 | 1/5 | 20 |
| | 13 | TTCGGCTAG | 9 | 1/5 | 20 |
| 3 | 1 | CGGGGC | 6 | 2/5 | 40 |
| | 3 | TCGGGGC | 7 | 2/5 | 40 |
| | 5 | TTCGGGGC | 8 | 2/5 | 40 |
| | 6 | ATTCGGGGC | 9 | 2/5 | 40 |
| 4 | 1 | GAAAAA | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |

| | | | | | |
|---|---|---|---|---|---|
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 5 | 1 | GAAAAT | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 6 | 1 | GAAAGC | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 7 | 1 | GAATAC | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 8 | 1 | GAATCC | 6 | 0/5 | 0 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 9 | 1 | GAACAA | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 10 | 1 | GAACAC | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |

| | | | | | |
|---|---|---|---|---|---|
| 11 | 1 | GAACTG | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 12 | 1 | GAACCA | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 13 | 1 | GAACGT | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 14 | 1 | GAACGG | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 15 | 1 | GAAGTG | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 16 | 1 | GAAGGA | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |
| | 7 | GAGAA | 5 | 1/5 | 20 |
| | 10 | AGAAC | 5 | 1/5 | 20 |
| | 11 | AGAAG | 5 | 1/5 | 20 |
| 17 | 1 | GAAGGG | 6 | 1/5 | 20 |
| | 2 | GAA | 3 | 13/5 | 260 |
| | 3 | AGAA | 4 | 2/5 | 40 |
| | 5 | TAGAA | 5 | 1/5 | 20 |

| | | | | |
|---|---|---|---|---|
| | 7 GAGAA | 5 | 1/5 | 20 |
| | 10 AGAAC | 5 | 1/5 | 20 |
| | 11 AGAAG | 5 | 1/5 | 20 |
| max | ACAGGA | 6 | 3/5 | 60 |

Največje število obiskov: 4

| # | v_i | kandidati | trojčki |
|---|---|---|---|
| 1 | 11 | 11 -> 8 | ACCATG |
| 2 | | 11 -> 14 | ACCAGT |
| 3 | | 11 -> 15 | ACCAGC |
| 4 | | 11 -> 39 | ACCCTC |
| 5 | | 11 -> 48 | ACCCGG |
| 6 | | 11 -> 49 | ACCGAA |
| 7 | | 11 -> 53 | ACCGTA |
| 8 | 53 | 53 -> 36 | GTACAG |
| 9 | | 53 -> 44 | GTACCG |
| 10 | 61 | 61 -> 4 | GGAAAG |
| 11 | | 61 -> 7 | GGAATC |
| 12 | | 61 -> 9 | GGAACA |
| 13 | | 61 -> 11 | GGAACC |
| 14 | | 61 -> 23 | GGATTC |
| 15 | | 61 -> 27 | GGATCC |
| 16 | | 61 -> 28 | GGATCG |
| 17 | | 61 -> 32 | GGATGG |
| 18 | | 61 -> 35 | GGACAC |
| 19 | | 61 -> 39 | GGACTC |
| 20 | | 61 -> 49 | GGAGAA |
| 21 | | 61 -> 52 | GGAGAG |
| 22 | | 61 -> 53 | GGAGTA |
| 23 | | 61 -> 54 | GGAGTT |
| 24 | | 61 -> 60 | GGAGCG |

Kandidati za motiv.

| C | # | motiv | W | našli | našli % |
|---|---|---|---|---|---|
| 1 | 1 | ACCATG | 6 | 1/5 | 20 |
| | 2 | ACC | 3 | 7/5 | 140 |
| | 3 | AACC | 4 | 2/5 | 40 |
| | 6 | CAACC | 5 | 1/5 | 20 |
| | 7 | GAACC | 5 | 1/5 | 20 |
| | 8 | AACCA | 5 | 2/5 | 40 |
| | 10 | AACCAT | 6 | 1/5 | 20 |
| | 12 | AACCAG | 6 | 1/5 | 20 |
| 2 | 1 | ACCAGT | 6 | 1/5 | 20 |
| | 2 | ACC | 3 | 7/5 | 140 |

| | | | | | |
|---|---|---|---|---|---|
| | 3 | AACC | 4 | 2/5 | 40 |
| | 6 | CAACC | 5 | 1/5 | 20 |
| | 7 | GAACC | 5 | 1/5 | 20 |
| | 8 | AACCA | 5 | 2/5 | 40 |
| | 10 | AACCAT | 6 | 1/5 | 20 |
| | 12 | AACCAG | 6 | 1/5 | 20 |
| 3 | 1 | ACCAGC | 6 | 1/5 | 20 |
| | 2 | ACC | 3 | 7/5 | 140 |
| | 3 | AACC | 4 | 2/5 | 40 |
| | 6 | CAACC | 5 | 1/5 | 20 |
| | 7 | GAACC | 5 | 1/5 | 20 |
| | 8 | AACCA | 5 | 2/5 | 40 |
| | 10 | AACCAT | 6 | 1/5 | 20 |
| | 12 | AACCAG | 6 | 1/5 | 20 |
| 4 | 1 | ACCCTC | 6 | 1/5 | 20 |
| | 2 | ACC | 3 | 7/5 | 140 |
| | 3 | AACC | 4 | 2/5 | 40 |
| | 6 | CAACC | 5 | 1/5 | 20 |
| | 7 | GAACC | 5 | 1/5 | 20 |
| | 8 | AACCA | 5 | 2/5 | 40 |
| | 10 | AACCAT | 6 | 1/5 | 20 |
| | 12 | AACCAG | 6 | 1/5 | 20 |
| 5 | 1 | ACCCGG | 6 | 1/5 | 20 |
| | 2 | ACC | 3 | 7/5 | 140 |
| | 3 | AACC | 4 | 2/5 | 40 |
| | 6 | CAACC | 5 | 1/5 | 20 |
| | 7 | GAACC | 5 | 1/5 | 20 |
| | 8 | AACCA | 5 | 2/5 | 40 |
| | 10 | AACCAT | 6 | 1/5 | 20 |
| | 12 | AACCAG | 6 | 1/5 | 20 |
| 6 | 1 | ACCGAA | 6 | 1/5 | 20 |
| | 2 | ACC | 3 | 7/5 | 140 |
| | 3 | AACC | 4 | 2/5 | 40 |
| | 6 | CAACC | 5 | 1/5 | 20 |
| | 7 | GAACC | 5 | 1/5 | 20 |
| | 8 | AACCA | 5 | 2/5 | 40 |
| | 10 | AACCAT | 6 | 1/5 | 20 |
| | 12 | AACCAG | 6 | 1/5 | 20 |
| 7 | 1 | ACCGTA | 6 | 1/5 | 20 |
| | 2 | ACC | 3 | 7/5 | 140 |
| | 3 | AACC | 4 | 2/5 | 40 |
| | 6 | CAACC | 5 | 1/5 | 20 |
| | 7 | GAACC | 5 | 1/5 | 20 |
| | 8 | AACCA | 5 | 2/5 | 40 |
| | 10 | AACCAT | 6 | 1/5 | 20 |
| | 12 | AACCAG | 6 | 1/5 | 20 |

| | | | | | |
|---|---|---|---|---|---|
| 8 | 1 | GTACAG | 6 | 2/5 | 40 |
| | 2 | AGTACAG | 7 | 1/5 | 20 |
| | 3 | TGTACAG | 7 | 1/5 | 20 |
| | 6 | GTACAGA | 7 | 1/5 | 20 |
| | 9 | GTACAGG | 7 | 1/5 | 20 |
| 9 | 1 | GTACCG | 6 | 2/5 | 40 |
| | 3 | TGTACCG | 7 | 1/5 | 20 |
| | 4 | CGTACCG | 7 | 1/5 | 20 |
| | 6 | GTACCGA | 7 | 1/5 | 20 |
| | 7 | GTACCGT | 7 | 1/5 | 20 |
| 10 | 1 | GGAAAG | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 11 | 1 | GGAATC | 6 | 0/5 | 0 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 12 | 1 | GGAACA | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |

| | | | | | |
|---|---|---|---|---|---|
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 13 | 1 | GGAACC | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 14 | 1 | GGATTC | 6 | 0/5 | 0 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 15 | 1 | GGATCC | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 16 | 1 | GGATCG | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |

| | | | | | |
|---|---|---|---|---|---|
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 17 | 1 | GGATGG | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 18 | 1 | GGACAC | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 19 | 1 | GGACTC | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 20 | 1 | GGAGAA | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |

| | | | | | |
|---|---|---|---|---|---|
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 21 | 1 | GGAGAG | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 22 | 1 | GGAGTA | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| 23 | 1 | GGAGTT | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |

| | | | | | |
|---|---|---|---|---|---|
| 24 | 1 | GGAGCG | 6 | 1/5 | 20 |
| | 2 | GGA | 3 | 12/5 | 240 |
| | 3 | AGGA | 4 | 5/5 | 100 |
| | 4 | AAGGA | 5 | 1/5 | 20 |
| | 5 | TAGGA | 5 | 1/5 | 20 |
| | 6 | CAGGA | 5 | 3/5 | 60 |
| | 7 | ACAGGA | 6 | 3/5 | 60 |
| | 8 | AACAGGA | 7 | 1/5 | 20 |
| | 9 | TACAGGA | 7 | 1/5 | 20 |
| | 11 | GACAGGA | 7 | 1/5 | 20 |
| | 12 | ACAGGAA | 7 | 1/5 | 20 |
| | 13 | ACAGGAT | 7 | 1/5 | 20 |
| | 14 | ACAGGAC | 7 | 1/5 | 20 |
| max | | ACAGGA | 6 | 3/5 | 60 |

Največje število obiskov: 3

| # | v_i | kandidati | trojčki |
|---|---|---|---|
| 1 | 6 | 6 -> 48 | ATTCGG |
| 2 | 21 | 21 -> 8 | TTAATG |
| 3 | | 21 -> 29 | TTATGA |
| 4 | | 21 -> 60 | TTAGCG |
| 5 | 29 | 29 -> 25 | TGATCA |
| 6 | 57 | 57 -> 24 | GCATTG |
| 7 | | 57 -> 30 | GCATGT |
| 8 | | 57 -> 58 | GCAGCT |
| 9 | 58 | 58 -> 6 | GCTATT |
| 10 | 63 | 63 -> 20 | GGCTAG |

Kandidati za motiv.

| C | # | motiv | W | našli | našli % |
|---|---|---|---|---|---|
| 1 | 1 | ATTCGG | 6 | 5/5 | 100 |
| | 3 | TATTCGG | 7 | 5/5 | 100 |
| | 6 | CTATTCGG | 8 | 5/5 | 100 |
| | **10** | **GCTATTCGG** | **9** | **5/5** | **100** |
| 2 | 1 | TTAATG | 6 | 1/5 | 20 |
| | 2 | TTA | 3 | 3/5 | 60 |
| | 5 | CTTA | 4 | 1/5 | 20 |
| | 6 | GTTA | 4 | 2/5 | 40 |
| | 9 | CGTTA | 5 | 2/5 | 40 |
| | 12 | CCGTTA | 6 | 1/5 | 20 |
| | 13 | GCGTTA | 6 | 1/5 | 20 |
| | 14 | CGTTAA | 6 | 1/5 | 20 |
| | 15 | CGTTAT | 6 | 1/5 | 20 |

| | | | | | |
|---|---|---|---|---|---|
| 3 | 1 | TTATGA | 6 | 1/5 | 20 |
| | 2 | TTA | 3 | 3/5 | 60 |
| | 5 | CTTA | 4 | 1/5 | 20 |
| | 6 | GTTA | 4 | 2/5 | 40 |
| | 9 | CGTTA | 5 | 2/5 | 40 |
| | 12 | CCGTTA | 6 | 1/5 | 20 |
| | 13 | GCGTTA | 6 | 1/5 | 20 |
| | 14 | CGTTAA | 6 | 1/5 | 20 |
| | 15 | CGTTAT | 6 | 1/5 | 20 |
| 4 | 1 | TTAGCG | 6 | 1/5 | 20 |
| | 2 | TTA | 3 | 3/5 | 60 |
| | 5 | CTTA | 4 | 1/5 | 20 |
| | 6 | GTTA | 4 | 2/5 | 40 |
| | 9 | CGTTA | 5 | 2/5 | 40 |
| | 12 | CCGTTA | 6 | 1/5 | 20 |
| | 13 | GCGTTA | 6 | 1/5 | 20 |
| | 14 | CGTTAA | 6 | 1/5 | 20 |
| | 15 | CGTTAT | 6 | 1/5 | 20 |
| 5 | 1 | TGATCA | 6 | 2/5 | 40 |
| | 2 | ATGATCA | 7 | 1/5 | 20 |
| | 4 | CTGATCA | 7 | 1/5 | 20 |
| | 6 | TGATCAA | 7 | 1/5 | 20 |
| | 7 | TGATCAT | 7 | 1/5 | 20 |
| 6 | 1 | GCATTG | 6 | 1/5 | 20 |
| | 2 | GCA | 3 | 3/5 | 60 |
| | 4 | TGCA | 4 | 2/5 | 40 |
| | 5 | ATGCA | 5 | 1/5 | 20 |
| | 8 | GTGCA | 5 | 1/5 | 20 |
| | 10 | TGCAT | 5 | 1/5 | 20 |
| | 12 | TGCAG | 5 | 1/5 | 20 |
| 7 | 1 | GCATGT | 6 | 1/5 | 20 |
| | 2 | GCA | 3 | 3/5 | 60 |
| | 4 | TGCA | 4 | 2/5 | 40 |
| | 5 | ATGCA | 5 | 1/5 | 20 |
| | 8 | GTGCA | 5 | 1/5 | 20 |
| | 10 | TGCAT | 5 | 1/5 | 20 |
| | 12 | TGCAG | 5 | 1/5 | 20 |
| 8 | 1 | GCAGCT | 6 | 1/5 | 20 |
| | 2 | GCA | 3 | 3/5 | 60 |
| | 4 | TGCA | 4 | 2/5 | 40 |
| | 5 | ATGCA | 5 | 1/5 | 20 |
| | 8 | GTGCA | 5 | 1/5 | 20 |
| | 10 | TGCAT | 5 | 1/5 | 20 |
| | 12 | TGCAG | 5 | 1/5 | 20 |
| 9 | 1 | GCTATT | 6 | 5/5 | 100 |
| | 2 | AGCTATT | 7 | 2/5 | 40 |

```
   3  AAGCTATT   8        1/5         20
   5  CAGCTATT   8        1/5         20
   9  AGCTATTC   8        2/5         40
   13 AGCTATTCG  9        2/5         40
-----------------------------------------
10 1  GGCTAG     6        3/5         60
   2  AGGCTAG    7        1/5         20
   3  TGGCTAG    7        1/5         20
   4  CGGCTAG    7        1/5         20
   6  GGCTAGA    7        2/5         40
   7  GGCTAGAA   8        1/5         20
   9  GGCTAGAC   8        1/5         20
-----------------------------------------
max   ATTCGG     6        5/5        100
```

Motivi prvega dela.

Motif #1: GCTATTCGG  9  5/5  1  na položajih:

```
1:  19
2:   7
3:  22
4:  14
5:  23
```

*  DRUGI DEL  *

---

Število posameznih trojčkov.

| trojček | # | trojček | # | trojček | # | trojček | # |
|---|---|---|---|---|---|---|---|
| AAA | 7 | AAT | 6 | AAC | 12 | AAG | 4 |
| ATA | 5 | ATT | 11 | ATC | 12 | ATG | 10 |
| ACA | 10 | ACT | 7 | ACC | 7 | ACG | 7 |
| AGA | 7 | AGT | 8 | AGC | 9 | AGG | 8 |
| TAA | 4 | TAT | 11 | TAC | 5 | TAG | 11 |
| TTA | 3 | TTT | 4 | TTC | 10 | TTG | 6 |
| TCA | 8 | TCT | 4 | TCC | 11 | TCG | 12 |
| TGA | 11 | TGT | 10 | TGC | 6 | TGG | 10 |
| CAA | 5 | CAT | 8 | CAC | 5 | CAG | 9 |
| CTA | 10 | CTT | 2 | CTC | 8 | CTG | 7 |
| CCA | 7 | CCT | 6 | CCC | 10 | CCG | 12 |
| CGA | 11 | CGT | 10 | CGC | 4 | CGG | 15 |
| GAA | 14 | GAT | 12 | GAC | 9 | GAG | 9 |
| GTA | 13 | GTT | 6 | GTC | 5 | GTG | 14 |
| GCA | 3 | GCT | 10 | GCC | 6 | GCG | 9 |
| GGA | 14 | GGT | 9 | GGC | 9 | GGG | 8 |

Frekvence trojčkov in psevdoštevila.

B: 2.23607
Vseh črk je: 535

| trojček | frekvenca | psevdokoda |
|---|---|---|
| AAA | 0.0130841 | 0.029257 |
| AAT | 0.011215 | 0.0250774 |
| AAC | 0.0224299 | 0.0501548 |
| AAG | 0.00747664 | 0.0167183 |
| ATA | 0.00934579 | 0.0208978 |
| ATT | 0.0205607 | 0.0459752 |
| ATC | 0.0224299 | 0.0501548 |
| ATG | 0.0186916 | 0.0417957 |
| ACA | 0.0186916 | 0.0417957 |
| ACT | 0.0130841 | 0.029257 |
| ACC | 0.0130841 | 0.029257 |
| ACG | 0.0130841 | 0.029257 |
| AGA | 0.0130841 | 0.029257 |
| AGT | 0.0149533 | 0.0334365 |
| AGC | 0.0168224 | 0.0376161 |
| AGG | 0.0149533 | 0.0334365 |
| TAA | 0.00747664 | 0.0167183 |
| TAT | 0.0205607 | 0.0459752 |
| TAC | 0.00934579 | 0.0208978 |
| TAG | 0.0205607 | 0.0459752 |
| TTA | 0.00560748 | 0.0125387 |
| TTT | 0.00747664 | 0.0167183 |
| TTC | 0.0186916 | 0.0417957 |
| TTG | 0.011215 | 0.0250774 |
| TCA | 0.0149533 | 0.0334365 |
| TCT | 0.00747664 | 0.0167183 |
| TCC | 0.0205607 | 0.0459752 |
| TCG | 0.0224299 | 0.0501548 |
| TGA | 0.0205607 | 0.0459752 |
| TGT | 0.0186916 | 0.0417957 |
| TGC | 0.011215 | 0.0250774 |
| TGG | 0.0186916 | 0.0417957 |
| CAA | 0.00934579 | 0.0208978 |
| CAT | 0.0149533 | 0.0334365 |
| CAC | 0.00934579 | 0.0208978 |
| CAG | 0.0168224 | 0.0376161 |
| CTA | 0.0186916 | 0.0417957 |
| CTT | 0.00373832 | 0.00835913 |
| CTC | 0.0149533 | 0.0334365 |
| CTG | 0.0130841 | 0.029257 |
| CCA | 0.0130841 | 0.029257 |
| CCT | 0.011215 | 0.0250774 |
| CCC | 0.0186916 | 0.0417957 |
| CCG | 0.0224299 | 0.0501548 |
| CGA | 0.0205607 | 0.0459752 |

```
CGT     0.0186916   0.0417957
CGC     0.00747664  0.0167183
CGG     0.0280374   0.0626935
GAA     0.0261682   0.0585139
GAT     0.0224299   0.0501548
GAC     0.0168224   0.0376161
GAG     0.0168224   0.0376161
GTA     0.0242991   0.0543344
GTT     0.011215    0.0250774
GTC     0.00934579  0.0208978
GTG     0.0261682   0.0585139
GCA     0.00560748  0.0125387
GCT     0.0186916   0.0417957
GCC     0.011215    0.0250774
GCG     0.0168224   0.0376161
GGA     0.0261682   0.0585139
GGT     0.0168224   0.0376161
GGC     0.0168224   0.0376161
GGG     0.0149533   0.0334365

Dolžine kodiranih sekvenc.
sek1:  308
sek2:  307
sek3:  304
sek4:  303
sek5:  306


Glavna zanka.
Število iteracij: 100


Vse pojavitve motiva 1 so bile določene v PRVEM DELU.


------------------------------------------------------------
                            Motiv 1
------------------------------------------------------------

Rešitev.

Število trojčkov: 7
Dolžina motiva: 9


sek  motiv      položaji
----------------------------
1    GCTATTCGG  19
2    GCTATTCGG  7
3    GCTATTCGG  22
4    GCTATTCGG  14
5    GCTATTCGG  23
```

Matrika Q.

| pol# | A | T | C | G |
|---|---|---|---|---|
| 1 | 0.0881622 | 0.0835567 | 0.0861884 | **0.90245** |
| 2 | 0.0881622 | 0.0835567 | **0.887976** | 0.100663 |
| 3 | 0.0881622 | **0.885344** | 0.0861884 | 0.100663 |
| 4 | **0.889949** | 0.0835567 | 0.0861884 | 0.100663 |
| 5 | 0.0881622 | **0.885344** | 0.0861884 | 0.100663 |
| 6 | 0.0881622 | **0.885344** | 0.0861884 | 0.100663 |
| 7 | 0.0881622 | 0.0835567 | **0.887976** | 0.100663 |
| 8 | 0.0881622 | 0.0835567 | 0.0861884 | **0.90245** |
| 9 | 0.0881622 | 0.0835567 | 0.0861884 | **0.90245** |

Frekvence ozadja:

0.257946 0.22404 0.241993 0.276021

Motiv iz matrike Q (>0.85):

GCTATTCGG

Presečni motiv:

GCTATTCGG

Čas izvršitve: 0 h 0 min 1 s (250 clicks).

# B. Primerjava

Martin Tompa je s sodelavci primerjal uspešnost trinajst različnih metod za iskanje motivov v DNA sekvencah [48]. Primerjava je bila narejena na enotni podatkovni bazi, ki vključuje tudi realne sekvence človeka, miši, muhe in glive. Realne podatkovne množice teh organizmov smo uporabili za preverjanje algoritma *GraphGibbs*.

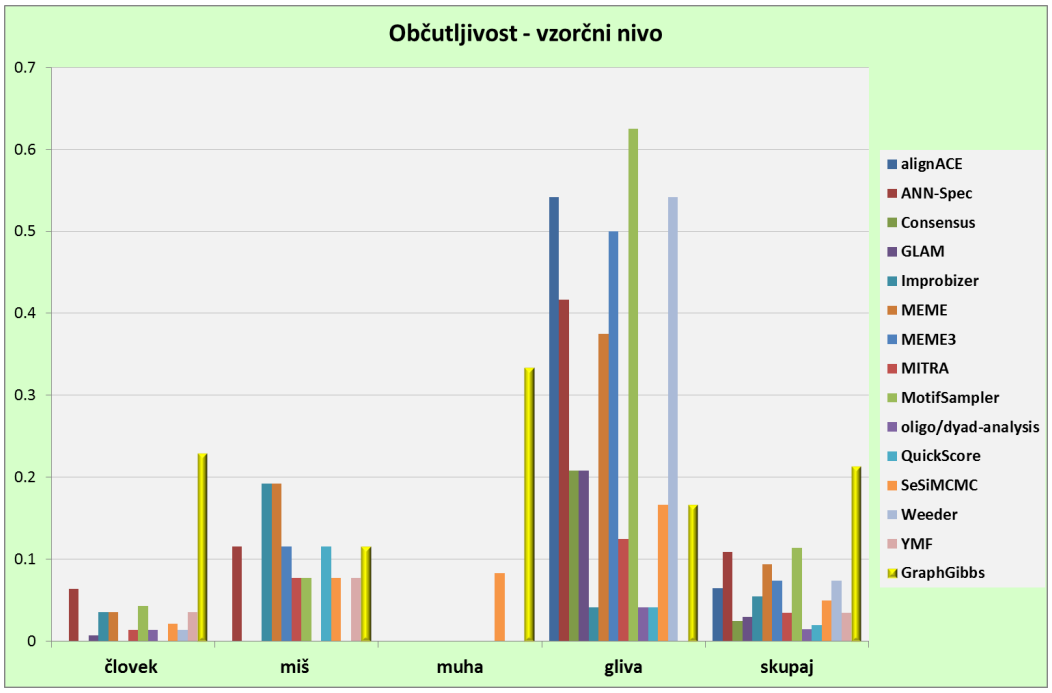
Na slikah B.1 in B.2 je grafično prikazana uspešnost trinajst različnih algoritmov in algoritma *GraphGibbs* na izbranih realnih množicah. Tompa in sodelavci so za primerjavo izbrali 9 množic sekvenc človeka, 4 množice sekvenc miši, 2 množici sekvenc muhe in 3 množice sekvenc glive. Na teh množicah lahko primerjamo tudi uspešnost algoritma *GraphGibbs*. Pri vsakem algoritmu je bil izbran najboljši rezultat oziroma najboljše ujemanje med znano in najdeno poravnavo. Vrednosti pozitivnih in negativnih ter lažno pozitivnih in lažno negativnih zadetkov od ostalih trinajst algoritmov je zbral Tompa s sodelavci. K tem rezultatom smo dodali še vrednosti posameznih zadetkov algoritma *GraphGibbs* pri najboljšem rezultatu v desetih ponovitvah.

Ker smo samo povzeli vrednosti zadetkov $TP$, $TN$, $FP$ in $FN$ na obeh nivojih, je primerjava uspešnosti preostalih metod z metodo *GraphGibbs* posredna in služi za okvirno rangiranje algoritma *GraphGibbs* med trinajst različnimi metodami za iskanje motivov v DNA sekvencah. Rezultate algoritma *GraphGibbs* smo že interpretirali v poglavjih 5 in 6. Za interpretacijo rezultatov ostalih metod pa usmerjamo bralca k članku [48].

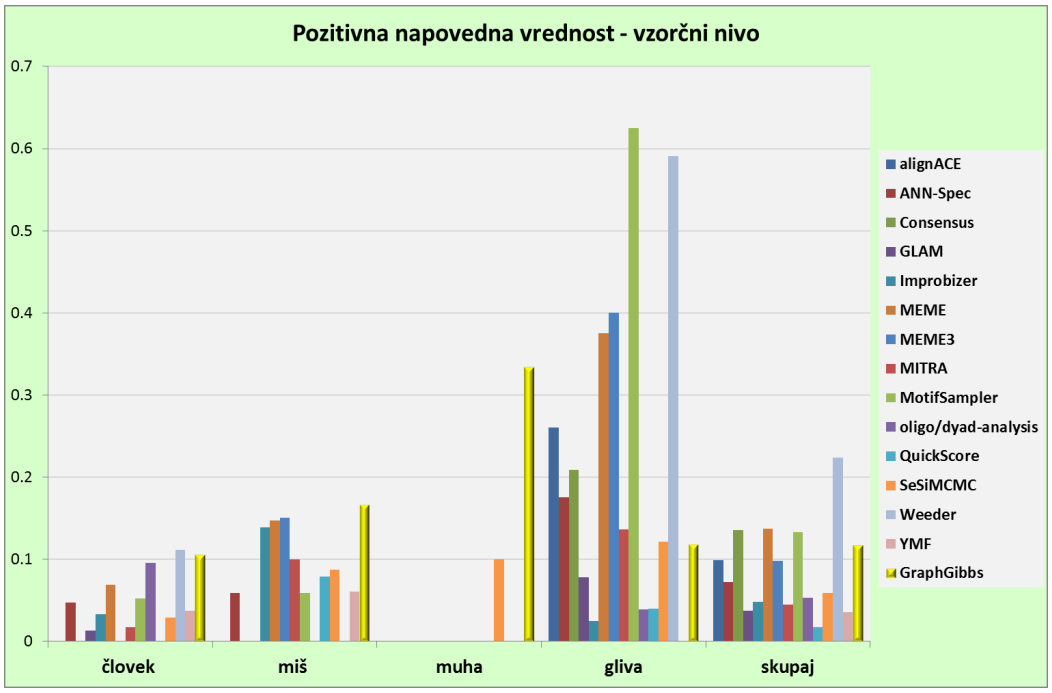
Na sliki B.1 (a) je prikazana občutljivost algoritmov na vzorčnem nivoju pri štirih organizmih. Zadnja kategorija predstavlja kumulativno uspešnost na vseh množicah v vzorcu. Iz grafa razberemo, da je občutljivost algoritma *GraphGibbs* najboljša na sekvencah človeka in muhe. Algoritem *GraphGibbs* je občutljiv tudi na sekvence miši. Najmanjša občutljivost se je izkazala pri sekvencah glive. Podoben primerjalni rang ima algoritem *GraphGibbs* tudi za pozitivno napovedno vrednost, ki je prikazana na grafu B.1 (b). Najvišjo PPV med vsemi algoritmi ima algoritem *GraphGibbs* pri množicah miši in muhe ter drugo najvišjo pri množicah sekvenc človeka. Pri množicah sekvenc glive je algoritem *GraphGibbs* deseti po vrsti glede na občutljivost in pozitivno napovedno vrednost. Kumulativno ima algoritem *GraphGibbs* najboljšo občutljivost in peto najboljšo PPV.

Na nukleotidnem nivoju smo naredili primerjavo vrednosti koeficienta uspešnosti,

slika B.2 (a), in korelacijskega koeficienta, slika B.2 (b). Algoritem *GraphGibbs* ima najvišje vrednosti obeh koeficientov med vsemi algoritmi pri dveh kategorijah: človeka in muhe. Pri obeh koeficientih ima tudi peto najvišjo vrednosti na sekvencah miši in deseto najvišjo vrednost na sekvencah glive. Kumulativna uspešnost na nukleotidnem nivoju algoritma *GraphGibbs* je peta najboljša v primerjavi z ostalimi algoritmi.

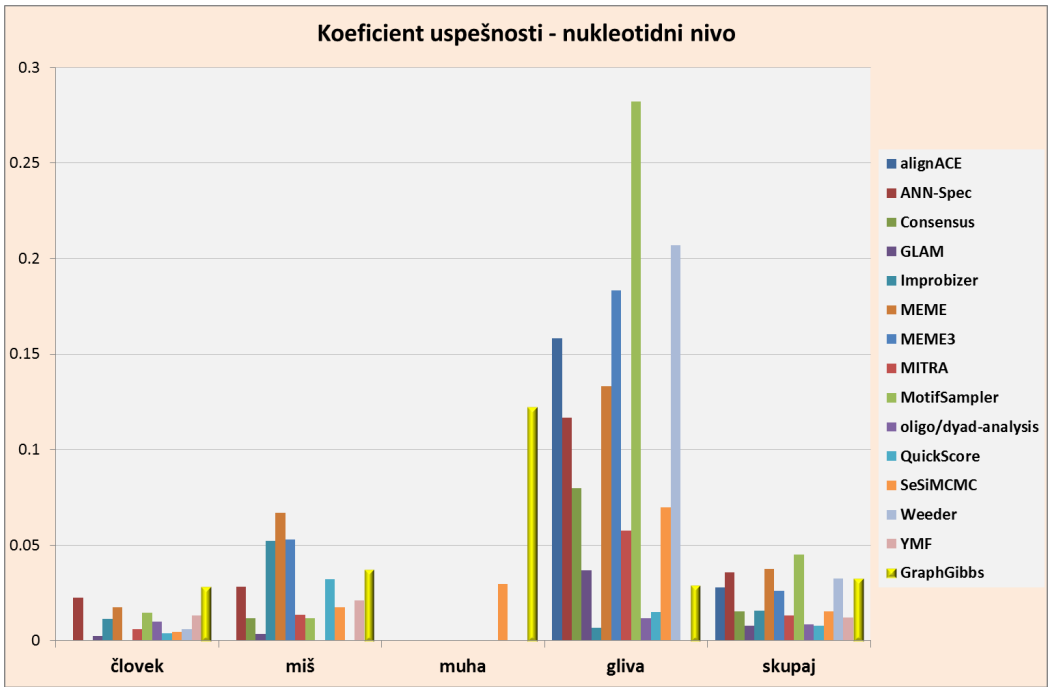



(a) Primerjava štirinajstih algoritmov za iskanje motivov - občutljivost na vzorčnem nivoju.

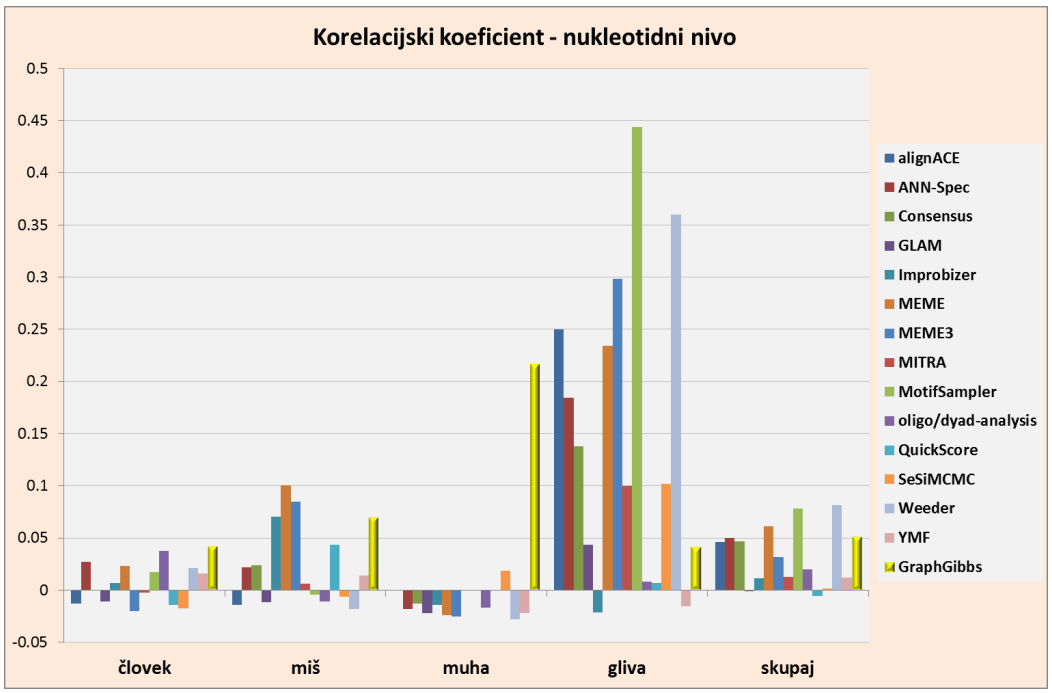



(b) Primerjava štirinajstih algoritmov za iskanje motivov - PPV na vzorčnem nivoju.

Slika B.1: Primerjava občutljivosti in pozitivne napovedne vrednosti na vzorčnem nivoju štirinajstih algoritmov za iskanje motivov v izbranih množicah DNA sekvenc štirih organizmov: človeka, miši, muhe in glive. Zadnja kategorija predstavlja kumulativno vrednost posamezne statistike po vseh izbranih množicah.




(a) Primerjava štirinajstih algoritmov za iskanje motivov - koeficient uspešnosti.

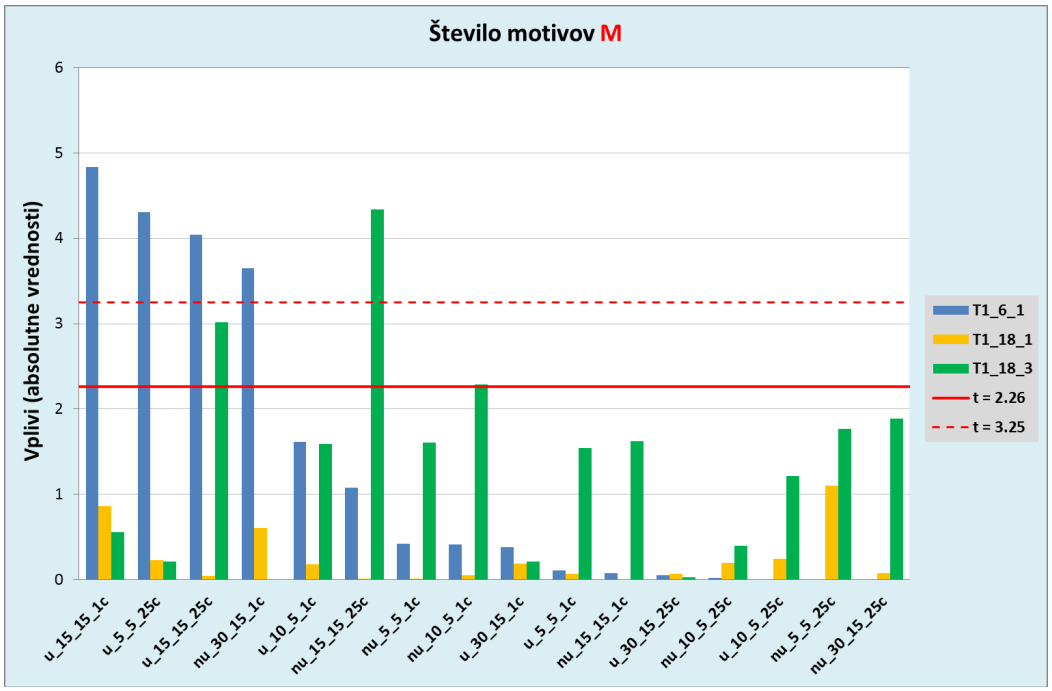



(b) Primerjava štirinajstih algoritmov za iskanje motivov - korelacijski koeficient.

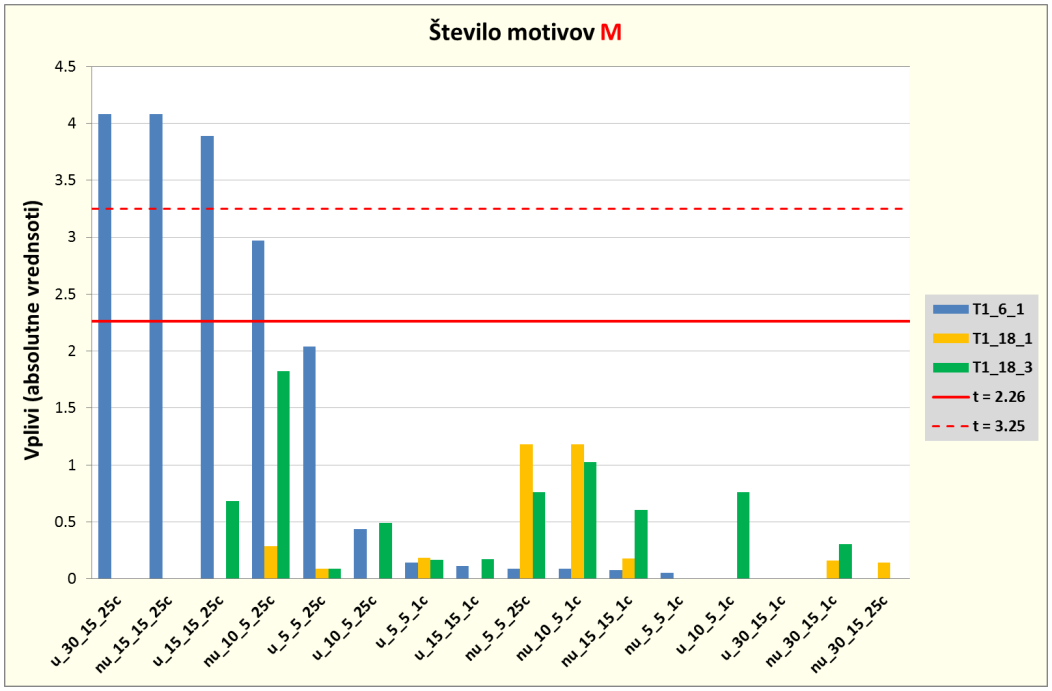
Slika B.2: Primerjava koeficientov uspešnosti in korelacije na nukleotidnem nivoju pri štirinajstih algoritmih za iskanje motivov v izbranih množicah DNA sekvenc štirih organizmov: človeka, miši, muhe in glive. Zadnja kategorija predstavlja kumulativno vrednost posamezne statistike po vseh izbranih množicah.

# C. Rezultati

## C.1 Načrt Plackett-Burman

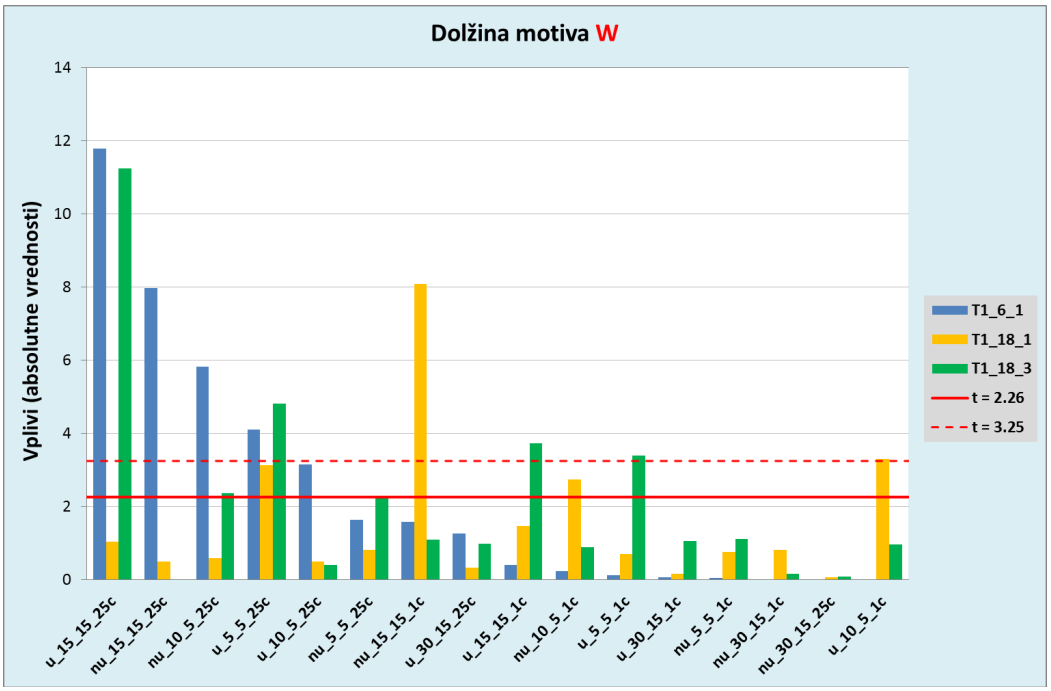



(a) Vplivi dejavnika $M$ na statistiko $I$.

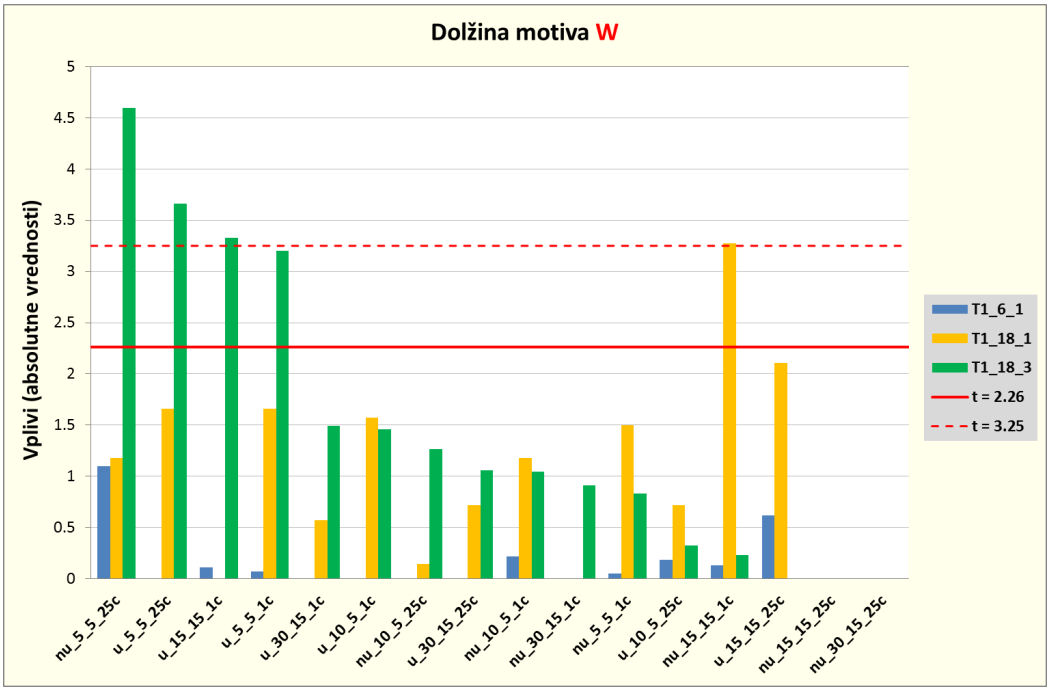



(b) Vplivi dejavnika $M$ na odstotek ujemanja $U$.

Slika C.1: Absolutne vrednosti vpliva dejavnika $M$ v posameznih množicah v vsaki skupini: $T1\_6\_1$, $T1\_18\_1$ in $T1\_18\_3$. Vpliv dejavnika merimo na: (a) statistiko $I$ in (b) odstotek ujemanja $U$.

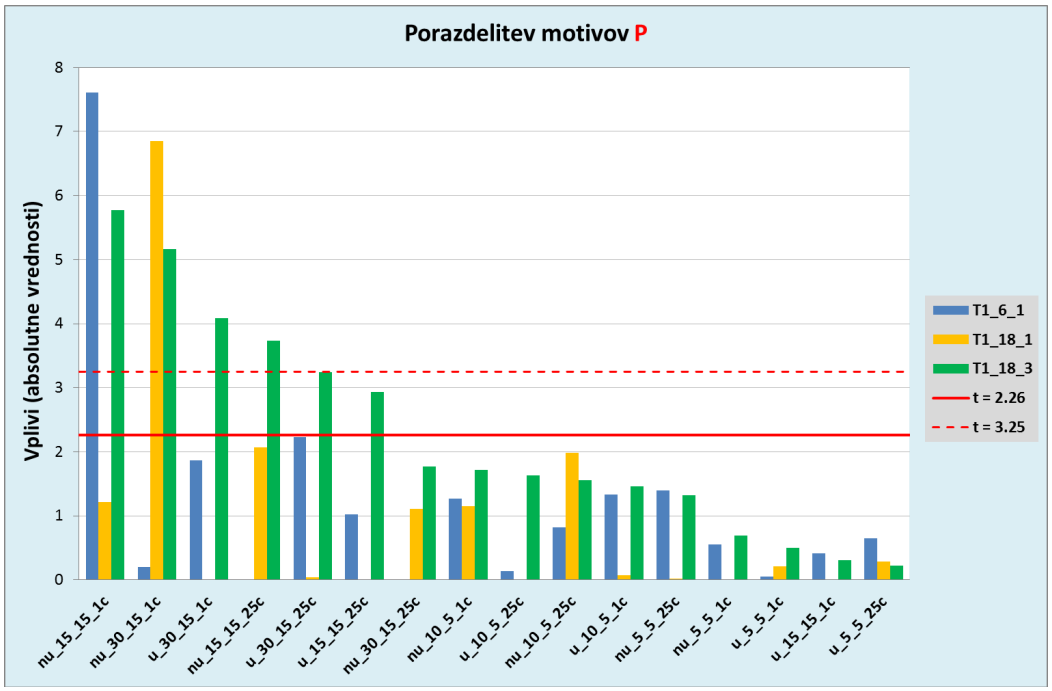



(a) Vplivi dejavnika $W$ na statistiko $I$.

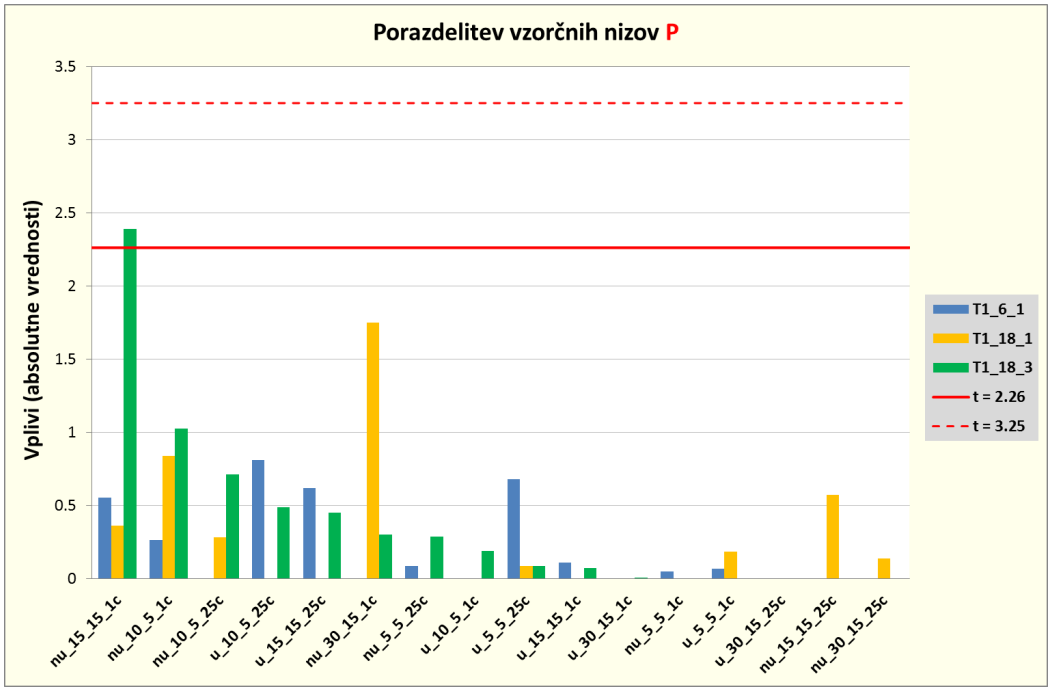



(b) Vplivi dejavnika $W$ na odstotek ujemanja $U$.

Slika C.2: Absolutne vrednosti vpliva dejavnika $W$ v posameznih množicah v vsaki skupini: $T1\_6\_1$, $T1\_18\_1$ in $T1\_18\_3$. Vpliv dejavnika merimo na: (a) statistiko $I$ in (b) odstotek ujemanja $U$.

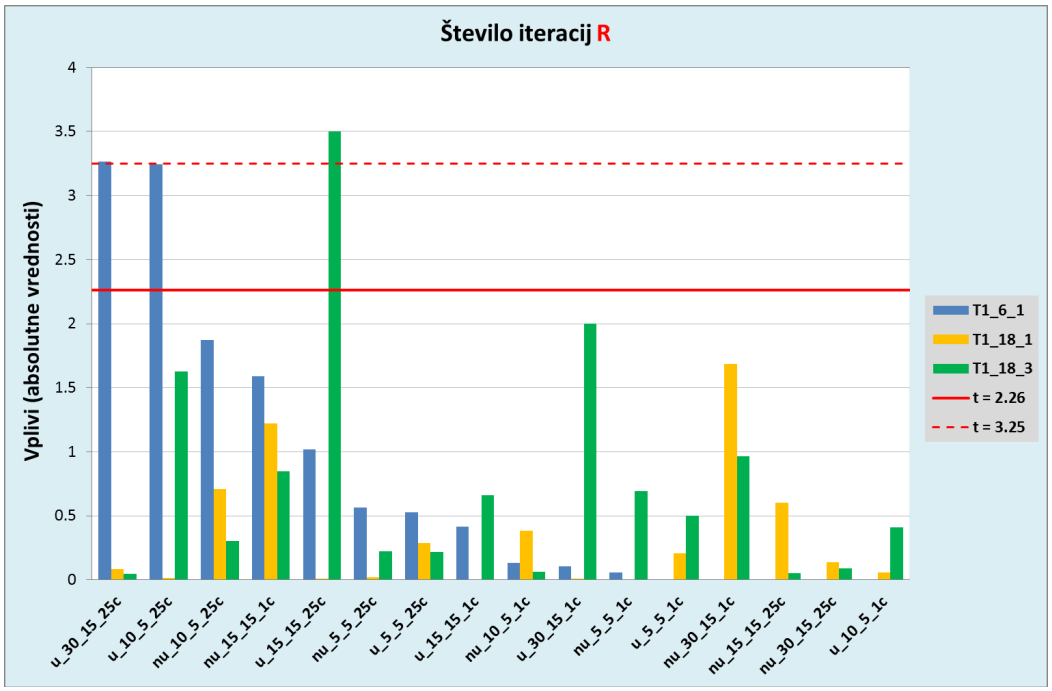



(a) Vplivi dejavnika $P$ na statistiko $I$.

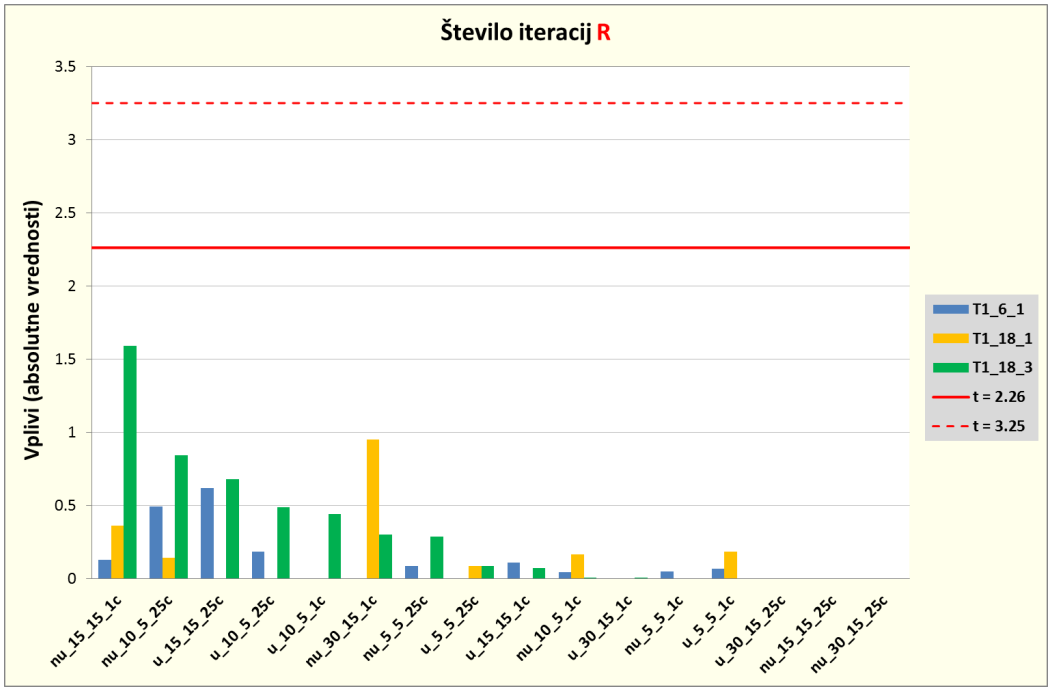



(b) Vplivi dejavnika $P$ na odstotek ujemanja $U$.

Slika C.3: Absolutne vrednosti vpliva dejavnika $P$ v posameznih množicah v vsaki skupini: $T1\_6\_1$, $T1\_18\_1$ in $T1\_18\_3$. Vpliv dejavnika merimo na: (a) statistiko $I$ in (b) odstotek ujemanja $U$.

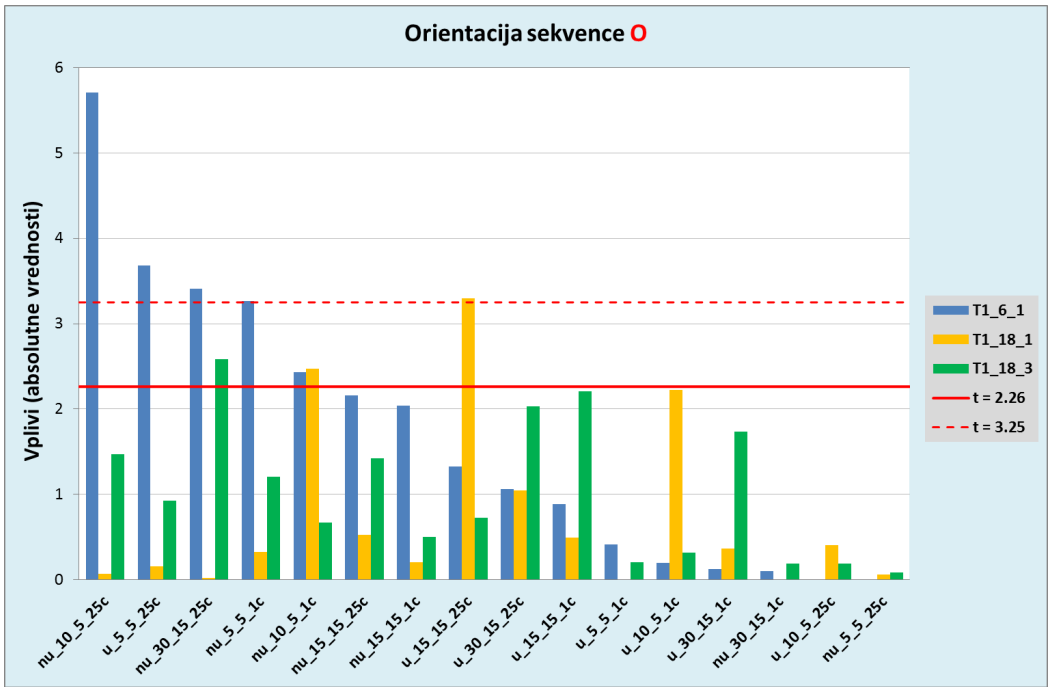



(a) Vplivi dejavnika $R$ na statistiko $I$.

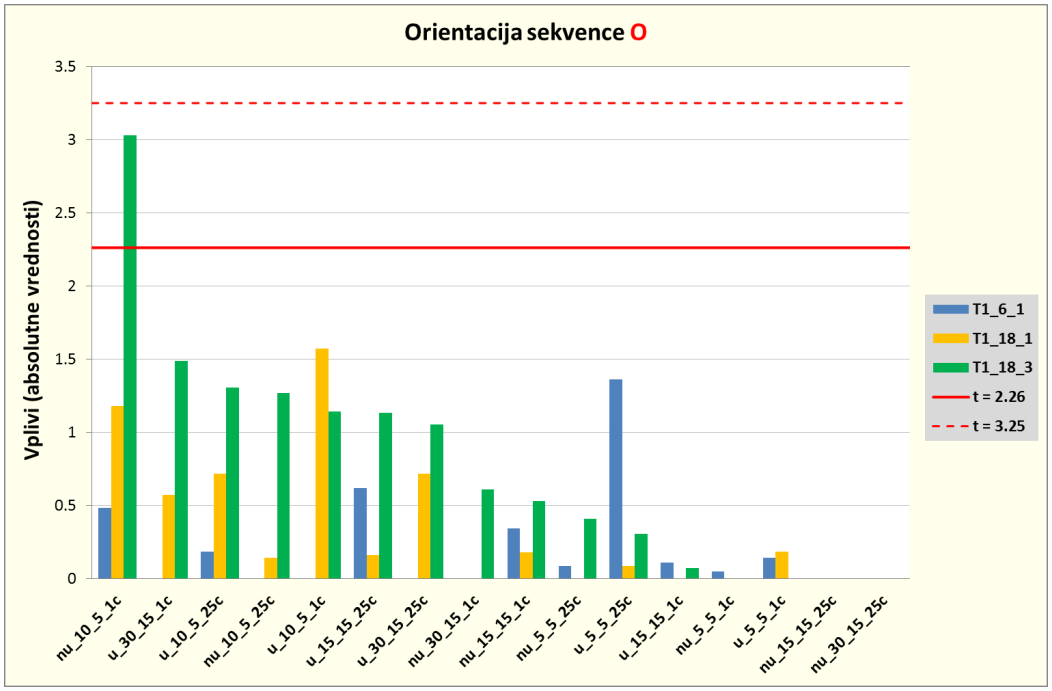



(b) Vplivi dejavnika $R$ na odstotek ujemanja $U$.

Slika C.4: Absolutne vrednosti vpliva dejavnika $R$ v posameznih množicah v vsaki skupini: $T1\_6\_1$, $T1\_18\_1$ in $T1\_18\_3$. Vpliv dejavnika merimo na: (a) statistiko $I$ in (b) odstotek ujemanja $U$.

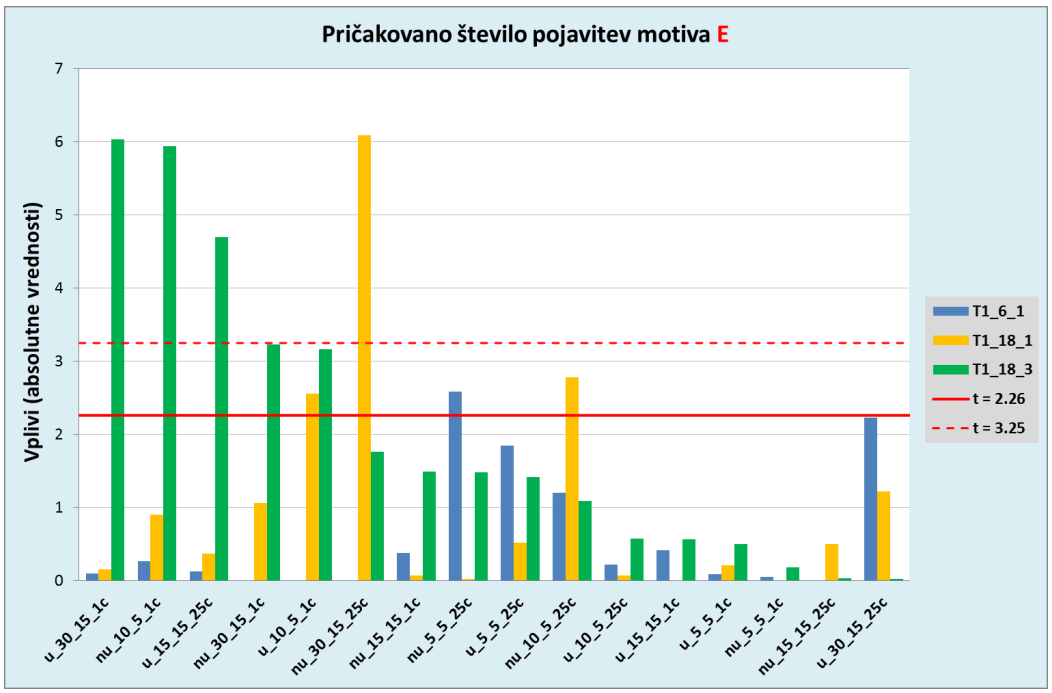



(a) Vplivi dejavnika $O$ na statistiko $I$.

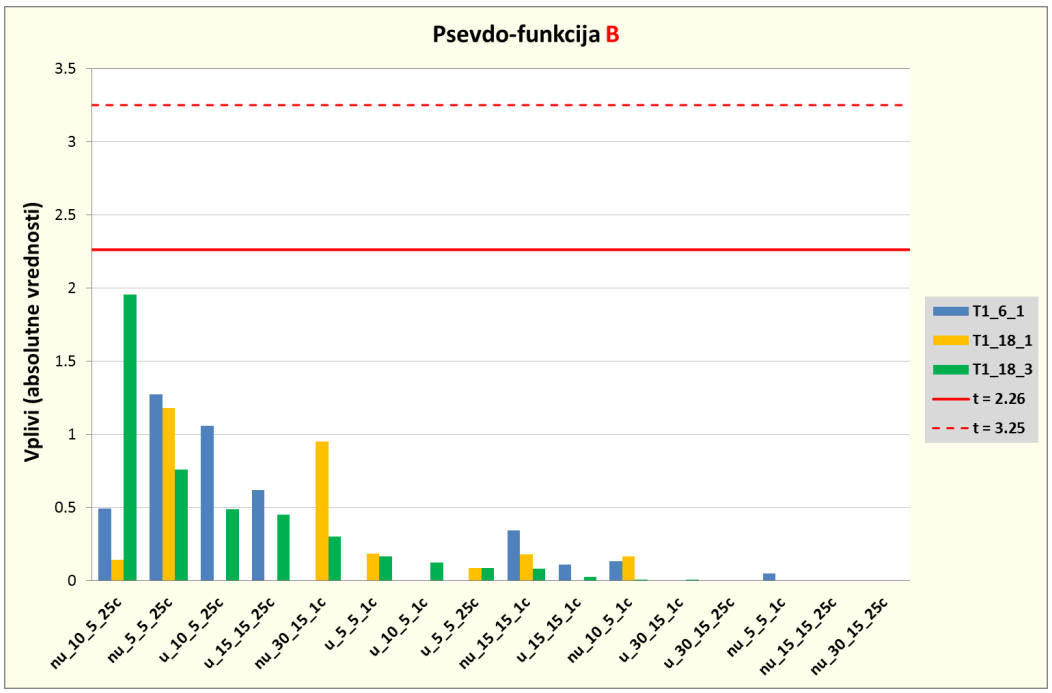



(b) Vplivi dejavnika $O$ na odstotek ujemanja $U$.

Slika C.5: Absolutne vrednosti vpliva dejavnika $O$ v posameznih množicah v vsaki skupini: $T1\_6\_1$, $T1\_18\_1$ in $T1\_18\_3$. Vpliv dejavnika merimo na: (a) statistiko $I$ in (b) odstotek ujemanja $U$.



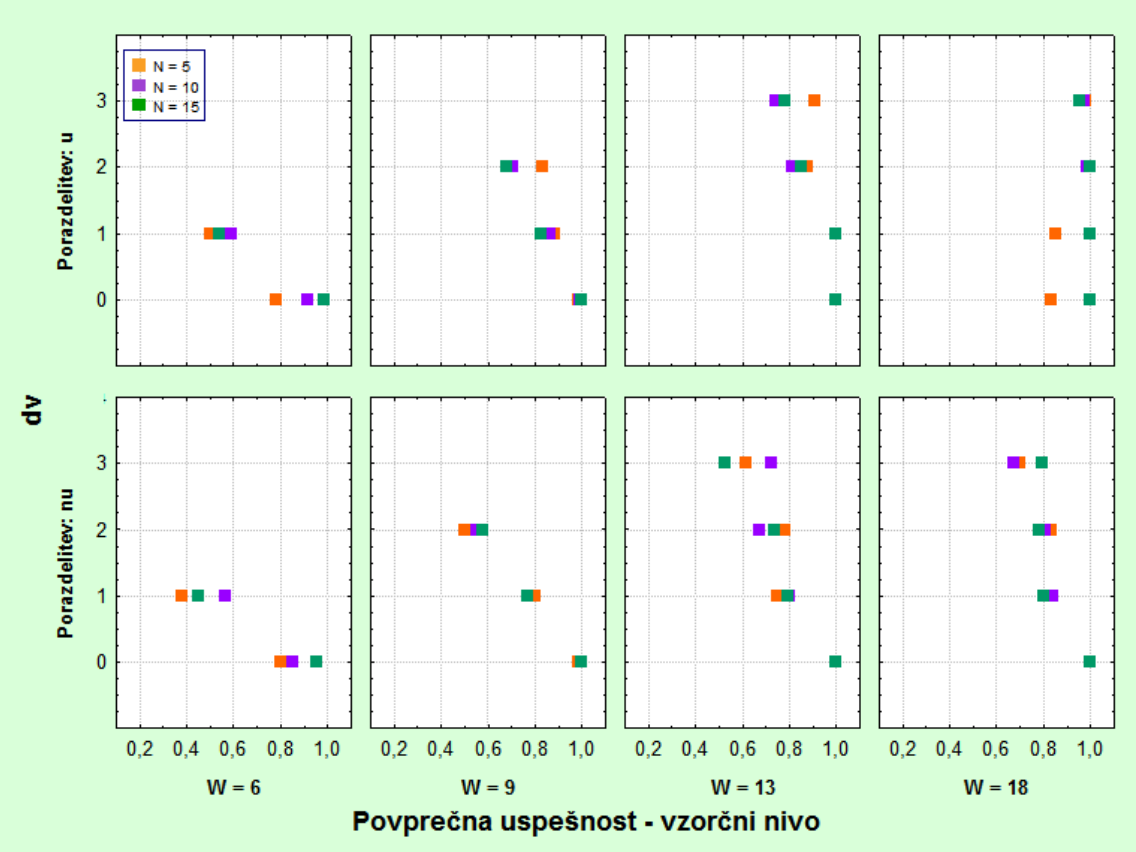

(a) Vplivi dejavnika $E$ na statistiko $I$.



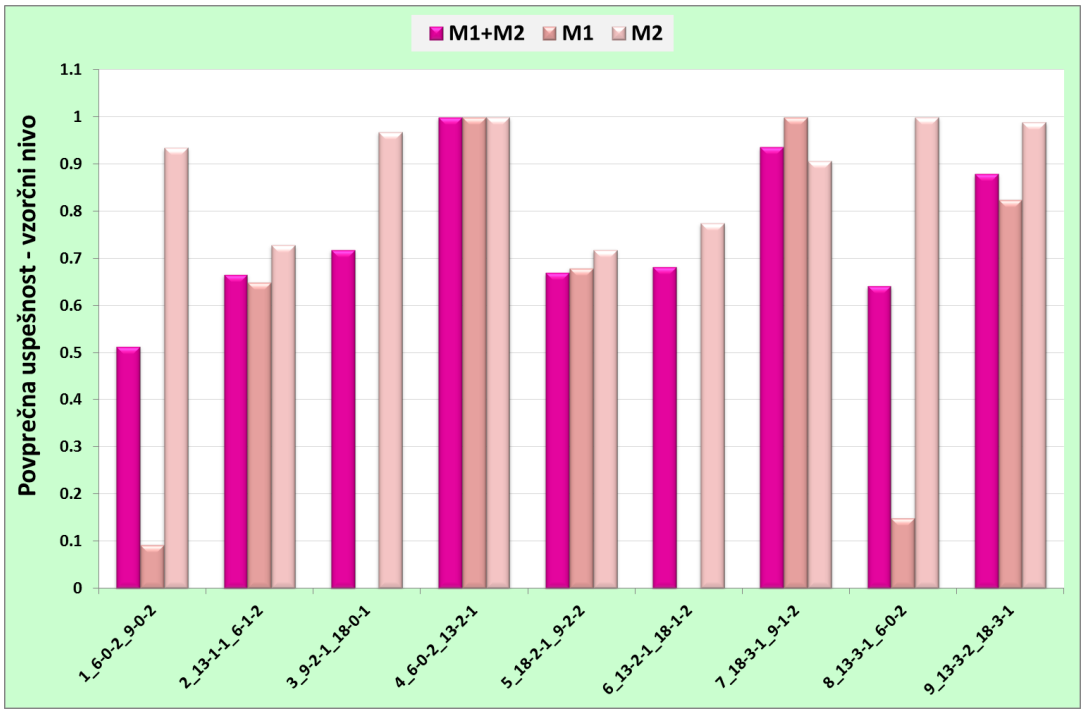

(b) Vplivi dejavnika $B$ na odstotek ujemanja $U$.

Slika C.6: Absolutne vrednosti vpliva dejavnika $E$ in $B$ v posameznih množicah v vsaki skupini: $T1\_6\_1$, $T1\_18\_1$ in $T1\_18\_3$. Grafa: (a) vpliv dejavnika $E$ na statistiko $I$ in (b) vpliv deajvnika $B$ na odstotek ujemanja $U$.

## C.2 Generirani podatki

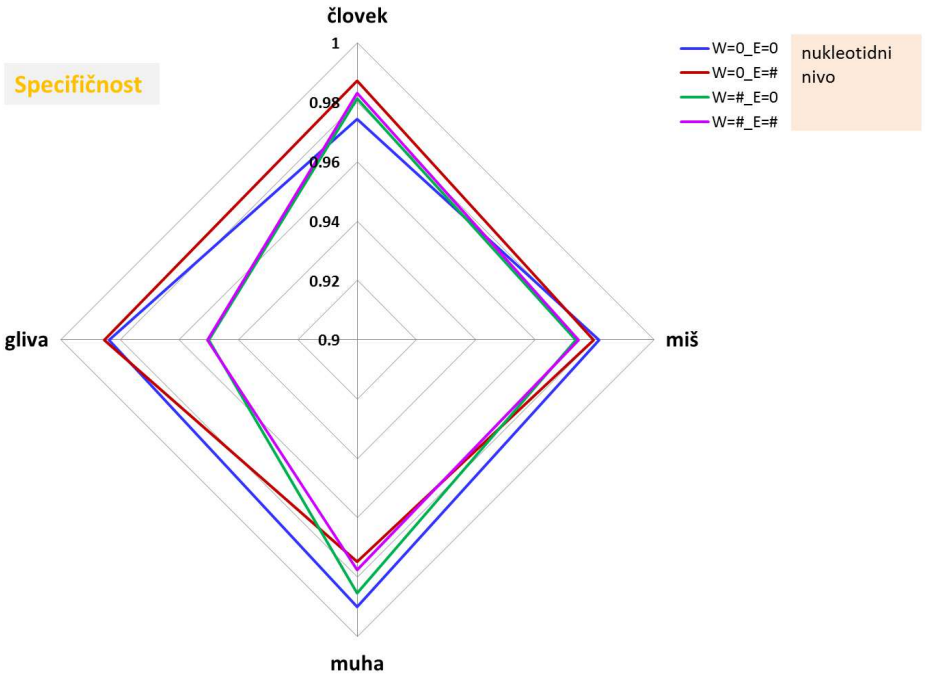



Slika C.7: Primerjalni grafi za prikaz povprečno uspešnost algoritma *GraphGibbs* na vzorčnem nivoju med dolžinami motiva $W \in \{6, 9, 13, 18\}$ glede na stopnjo variabilnosti $dv \in \{0, 1, 2, 3\}$, ki so ločeni glede na porazdelitev vzorčnih nizov $P \in \{u, nu\}$.

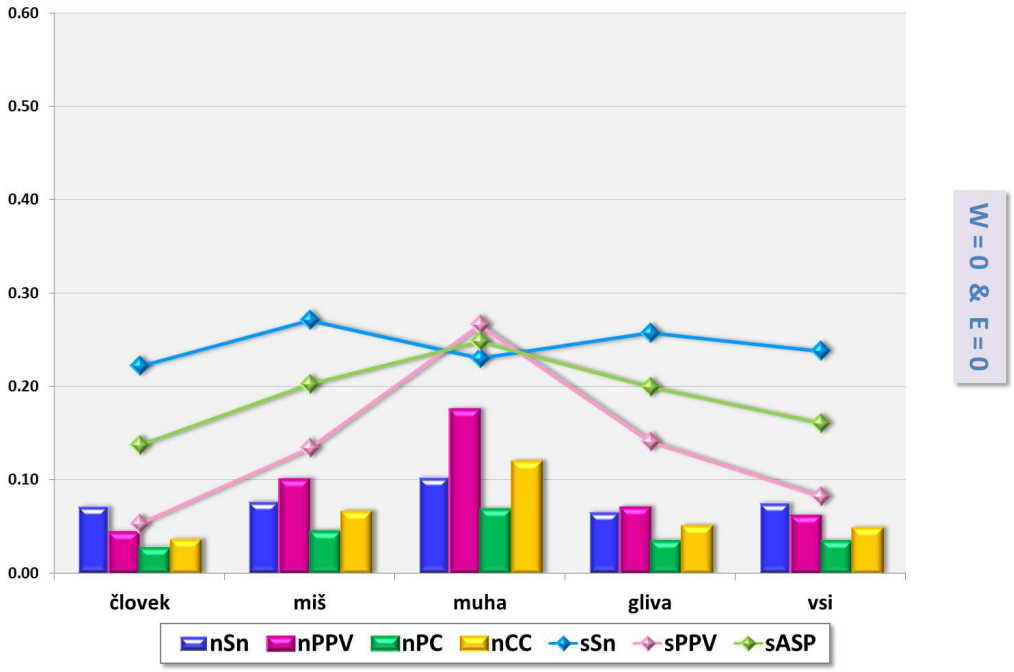



Slika C.8: Povprečna uspešnost algoritma *GraphGibbs* na vzorčnem nivoju za množice z dvema znanima motivoma.

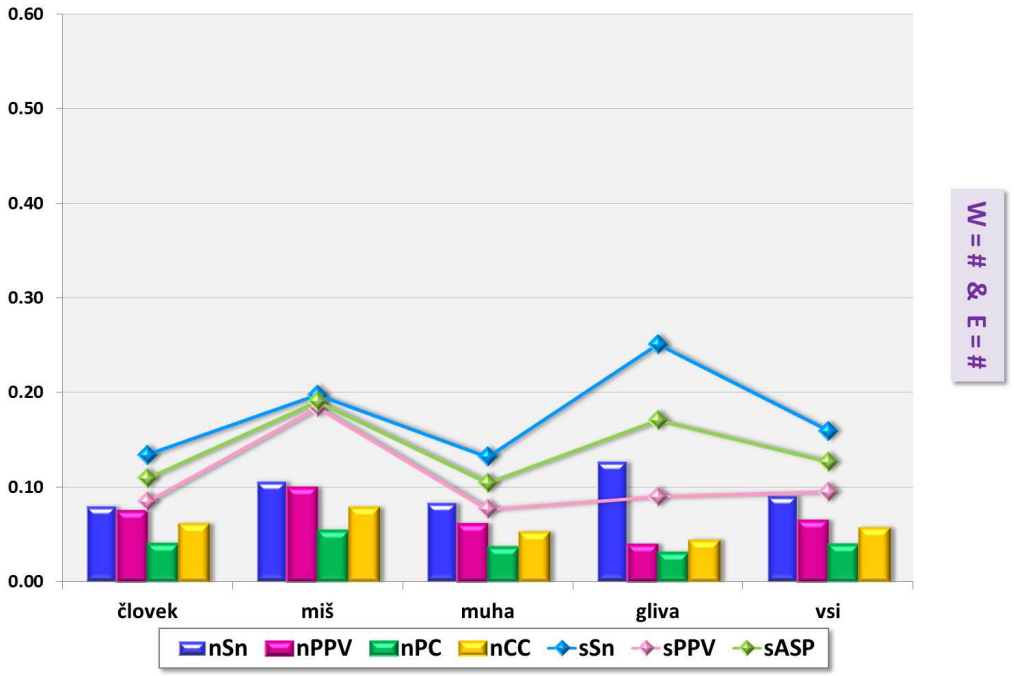
## C.3 Realni podatki




Slika C.9: Polarni graf specifičnosti algoritma *GraphGibbs* na nukleodtidnem nivoju za štiri skupine realnih množic pri štirih nastavitvah algoritma.



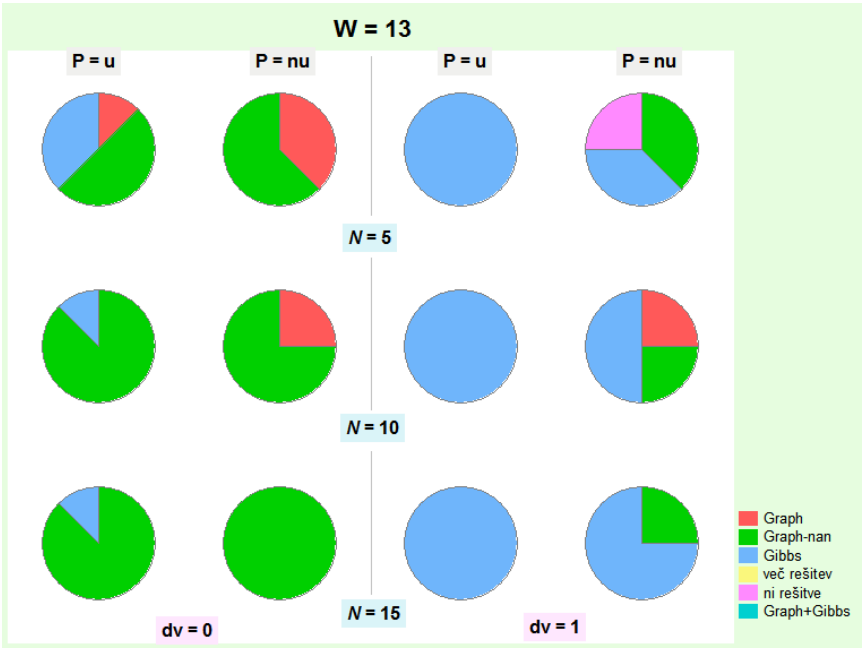

Slika C.10: Primerjava uspešnosti algoritma *GraphGibbs* pri prvi nastavitvi prostih parametrov z vrednostmi statistik nSn, nPPV, nPC in nCC na nukleotidnem nivoju (stolpci) ter vrednosti statistik sSn, sPPV in sASP na vzorčnem nivoju (črte).



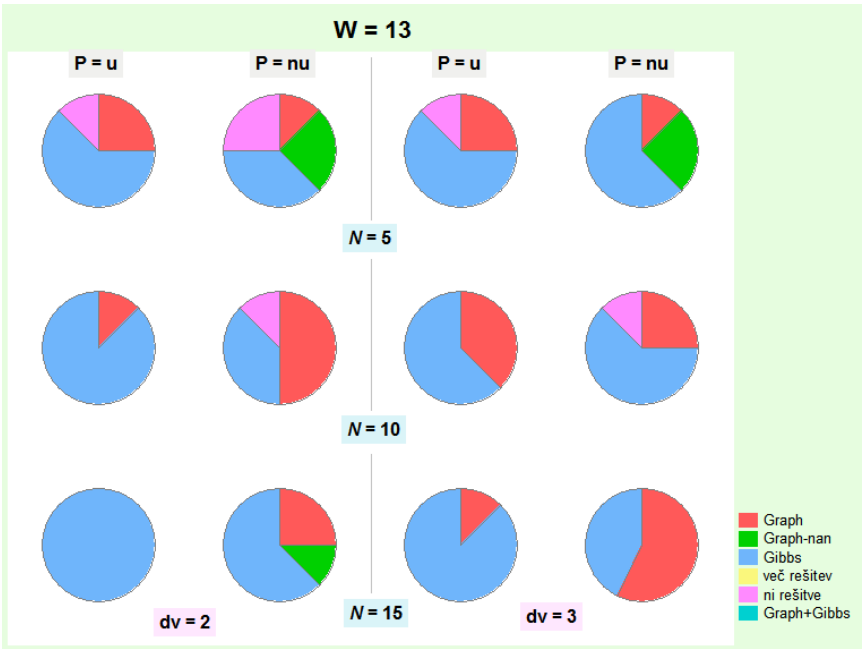

Slika C.11: Primerjava uspešnosti algoritma *GraphGibbs* pri četrti nastavitvi prostih parametrov z vrednostmi statistik nSn, nPPV, nPC in nCC na nukleotidnem nivoju (stolpci) ter vrednosti statistik sSn, sPPV in sASP na vzorčnem nivoju (črte).

## C.4 Analiza konvergence

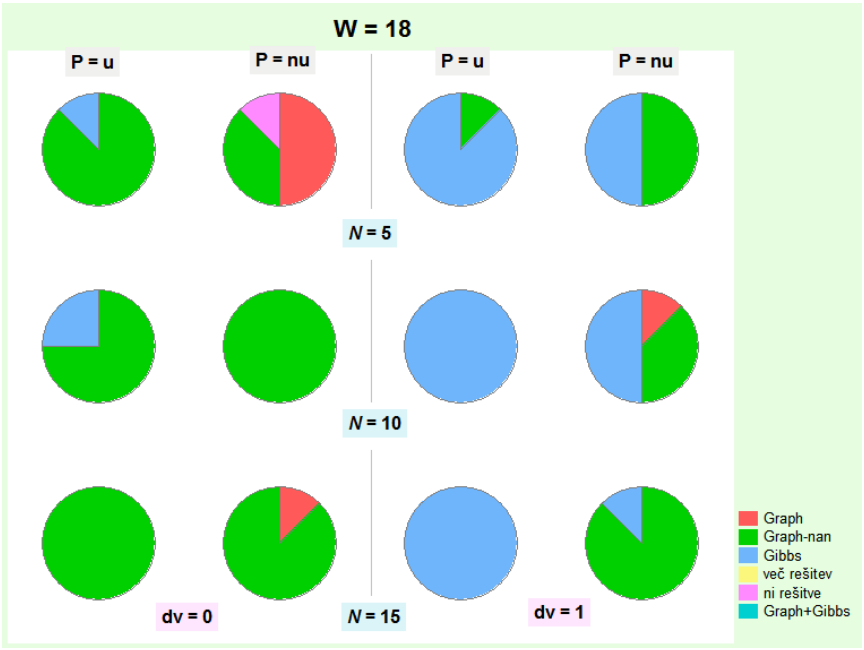

Slika C.12: Krožni diagram delitve množic z motivom dolžine $W = 6$ glede na način določitve rešitve algoritma *GraphGibbs*. V vrstah so prikazani diagrami glede na število sekvenc $N \in \{5, 10, 15\}$ v množicah. Po stopcih so pa v paru po porazdelitvi $P \in \{u, nu\}$ prikazani diagrami glede na stopnjo variabilnosti $dv \in \{0, 1, 2\}$.



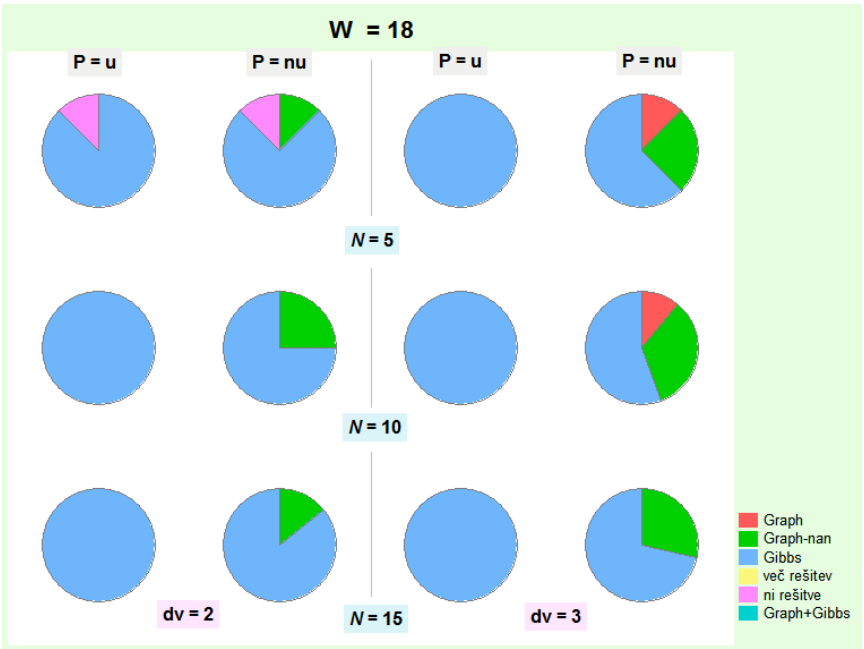

Slika C.13: Krožni diagram delitve množic z motivom dolžine $W = 13$ glede na način določitve rešitve algoritma *GraphGibbs*. V vrstah so prikazani diagrami glede na število sekvenc $N \in \{5, 10, 15\}$ v množicah. Po stopcih so pa v paru po porazdelitvi $P \in \{u, nu\}$ prikazani diagrami glede na stopnjo variabilnosti $dv \in \{0, 1\}$.




Slika C.14: Krožni diagram delitve množic z motivom dolžine $W = 13$ glede na način določitve rešitve algoritma *GraphGibbs*. V vrstah so prikazani diagrami glede na število sekvenc $N \in \{5, 10, 15\}$ v množicah. Po stopcih so pa v paru po porazdelitvi $P \in \{u, nu\}$ prikazani diagrami glede na stopnjo variabilnosti $dv \in \{2, 3\}$.

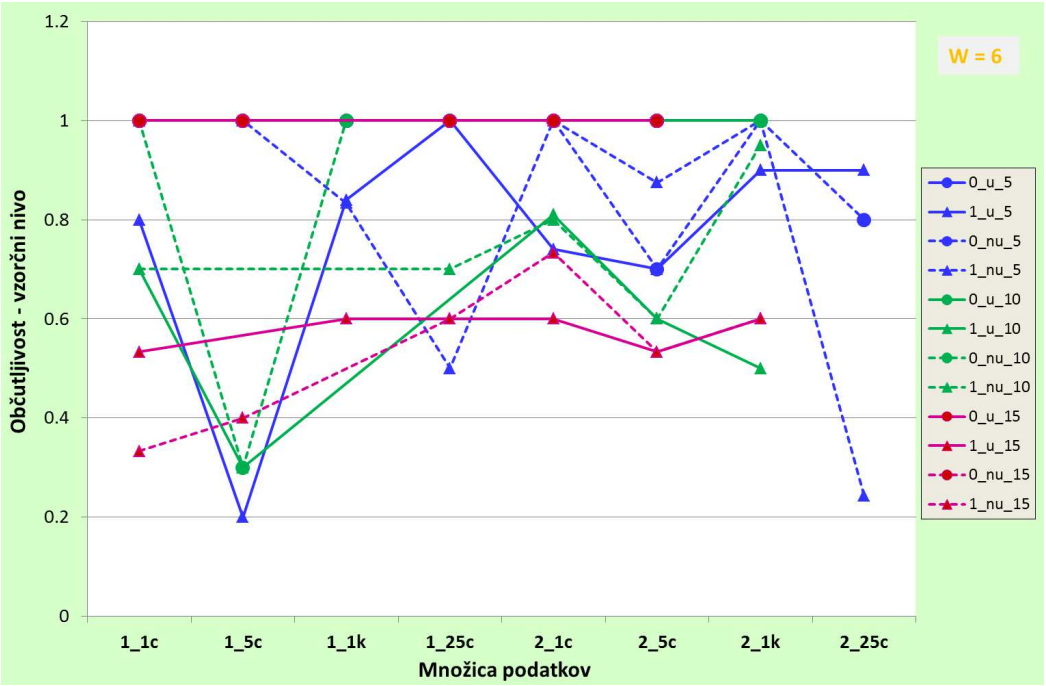



Slika C.15: Krožni diagram delitve množic z motivom dolžine $W = 18$ glede na način določitve rešitve algoritma *GraphGibbs*. V vrstah so prikazani diagrami glede na število sekvenc $N \in \{5, 10, 15\}$ v množicah. Po stopcih so pa v paru po porazdelitvi $P \in \{u, nu\}$ prikazani diagrami glede na stopnjo variabilnosti $dv = \{0, 1\}$.

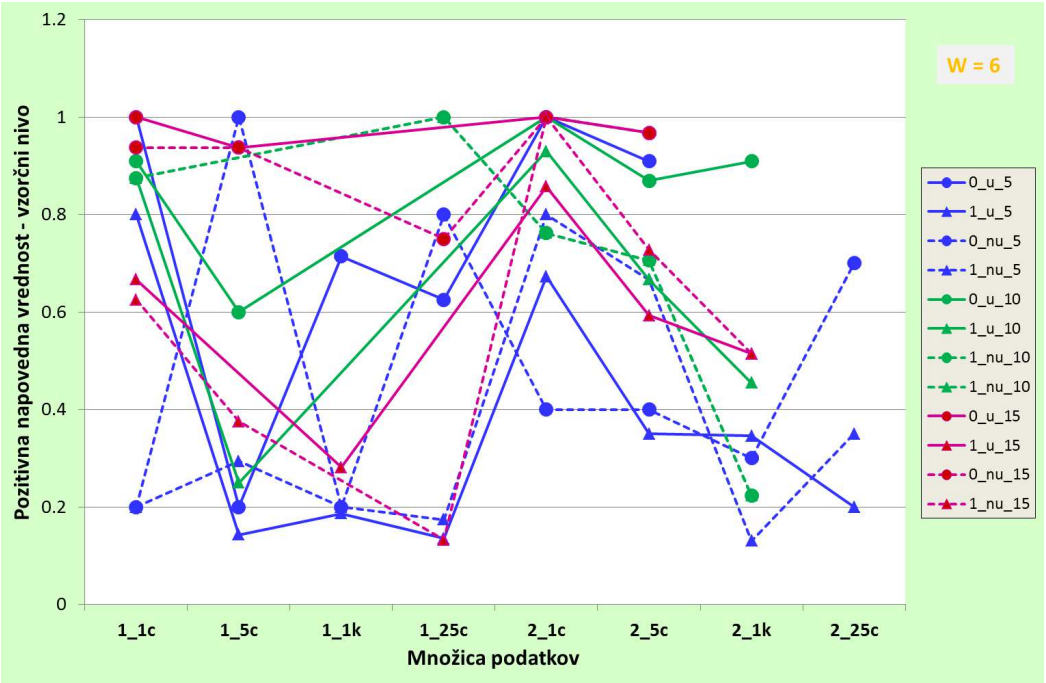



Slika C.16: Krožni diagram delitve množic z motivom dolžine $W = 18$ glede na način določitve rešitve algoritma *GraphGibbs*. V vrstah so prikazani diagrami glede na število sekvenc $N \in \{5, 10, 15\}$ v množicah. Po stopcih so pa v paru po porazdelitvi $P \in \{u, nu\}$ prikazani diagrami glede na stopnjo variabilnosti $dv \in \{2, 3\}$.

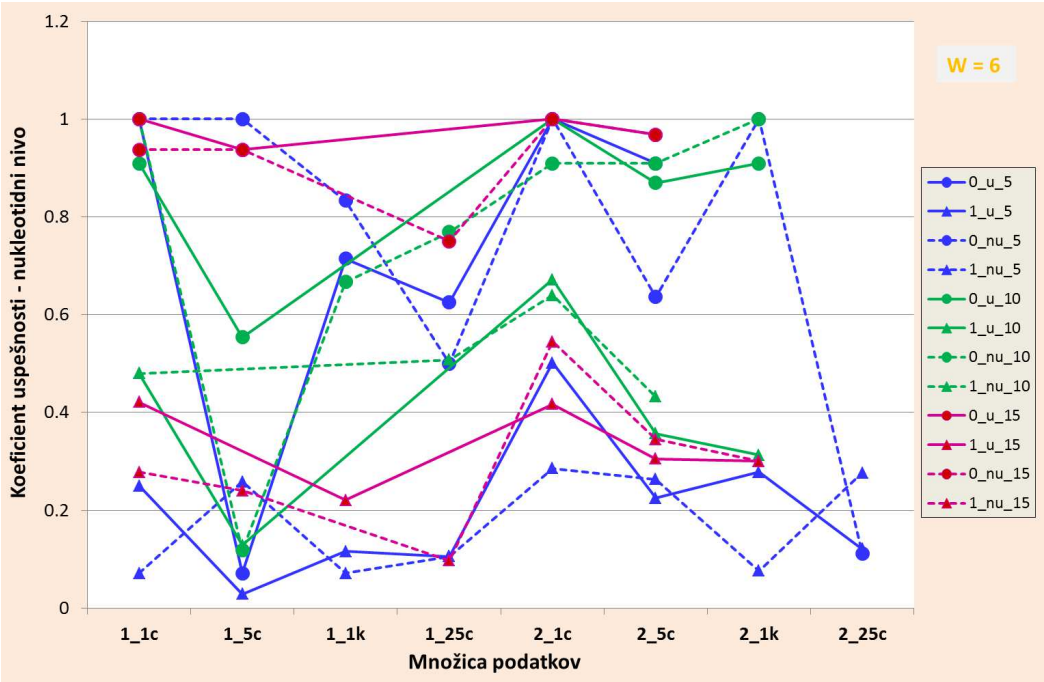
## C.5 Razprava



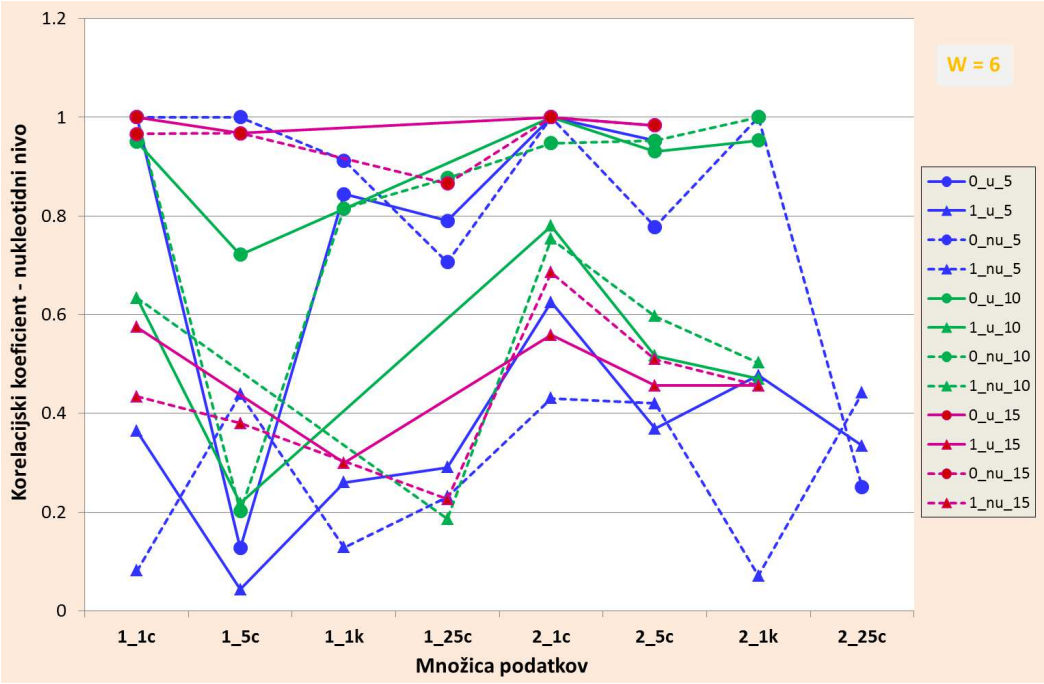

Slika C.17: Občutljivost algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 6$, vzorčni nivo.



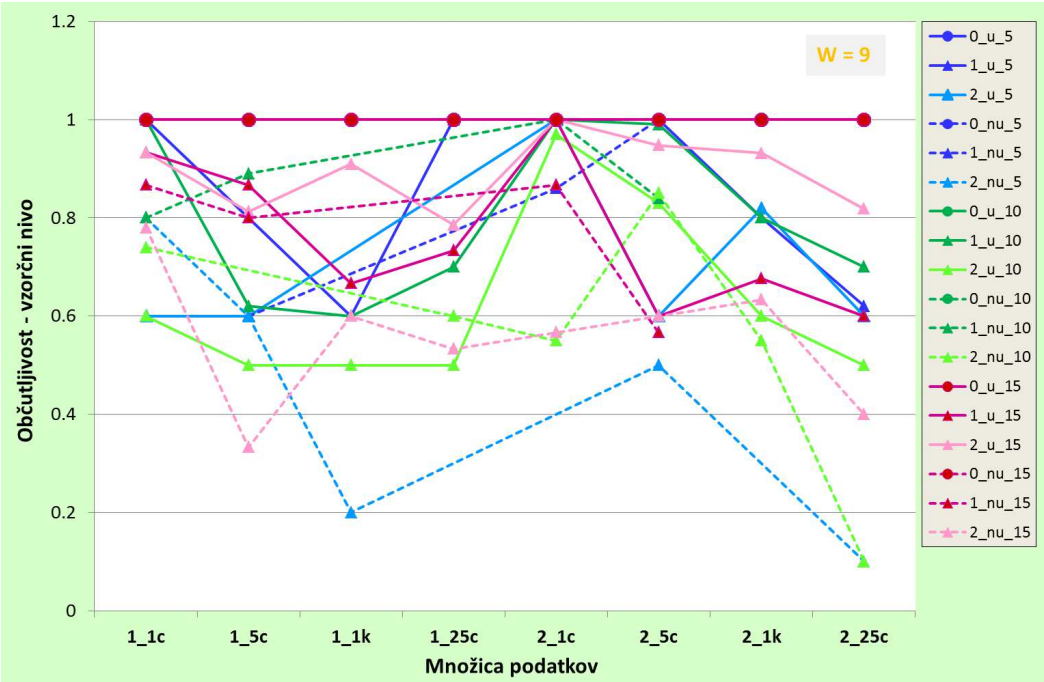

Slika C.18: Pozitivna napovedna vrednost algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 6$, vzorčni nivo.



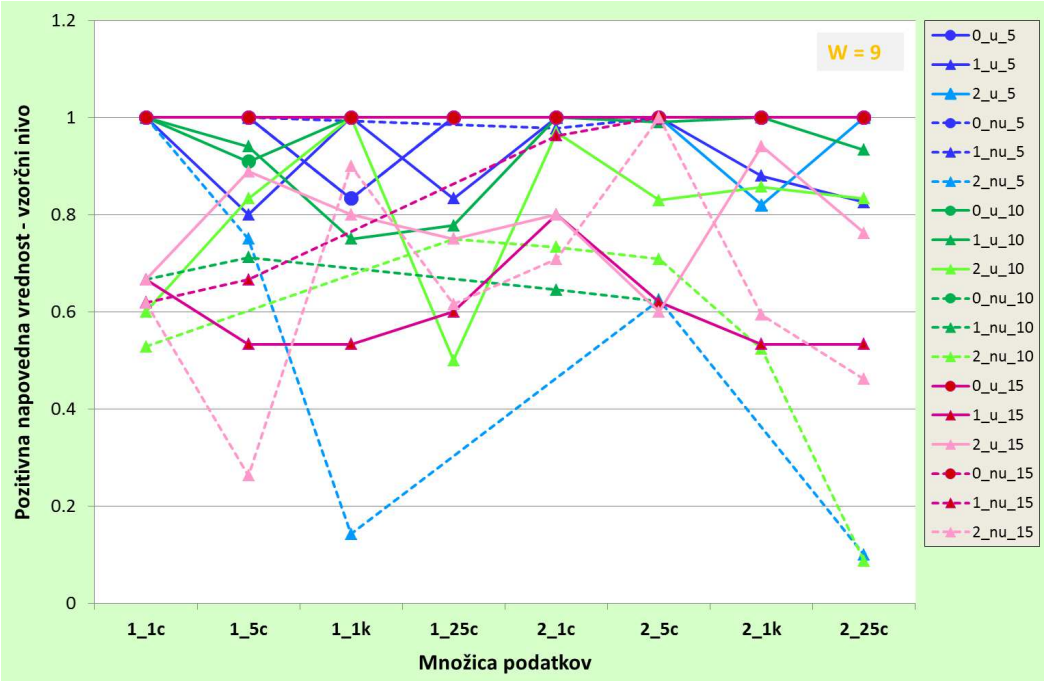

Slika C.19: Koeficient uspešnosti algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 6$, nukleotidni nivo.



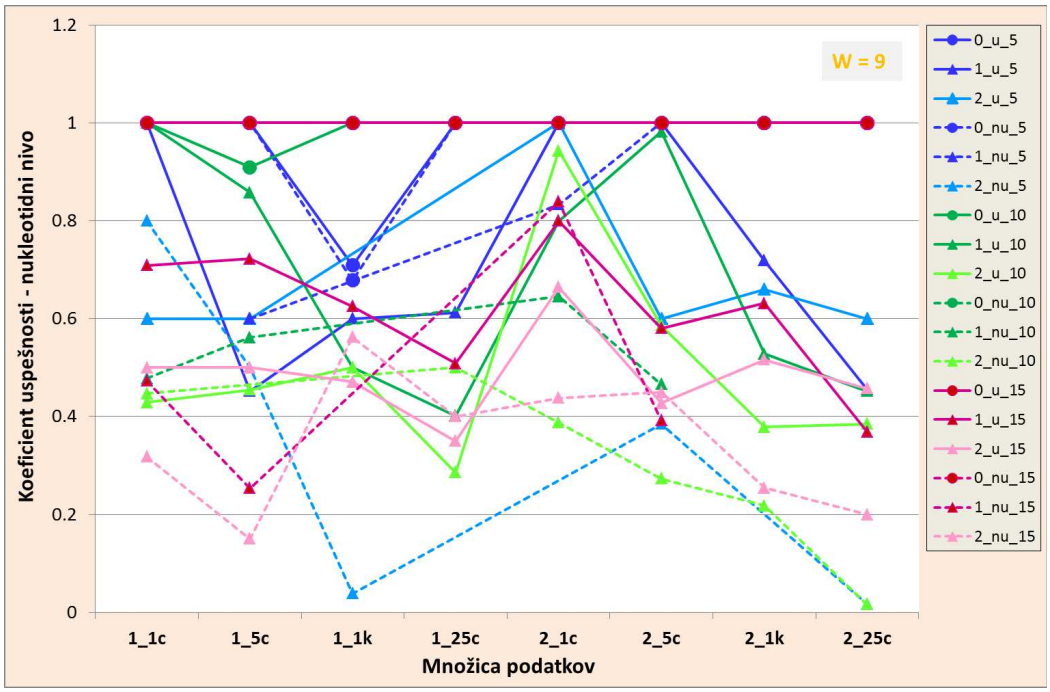

Slika C.20: Korelacijski koeficient algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 6$, nukleotidni nivo.



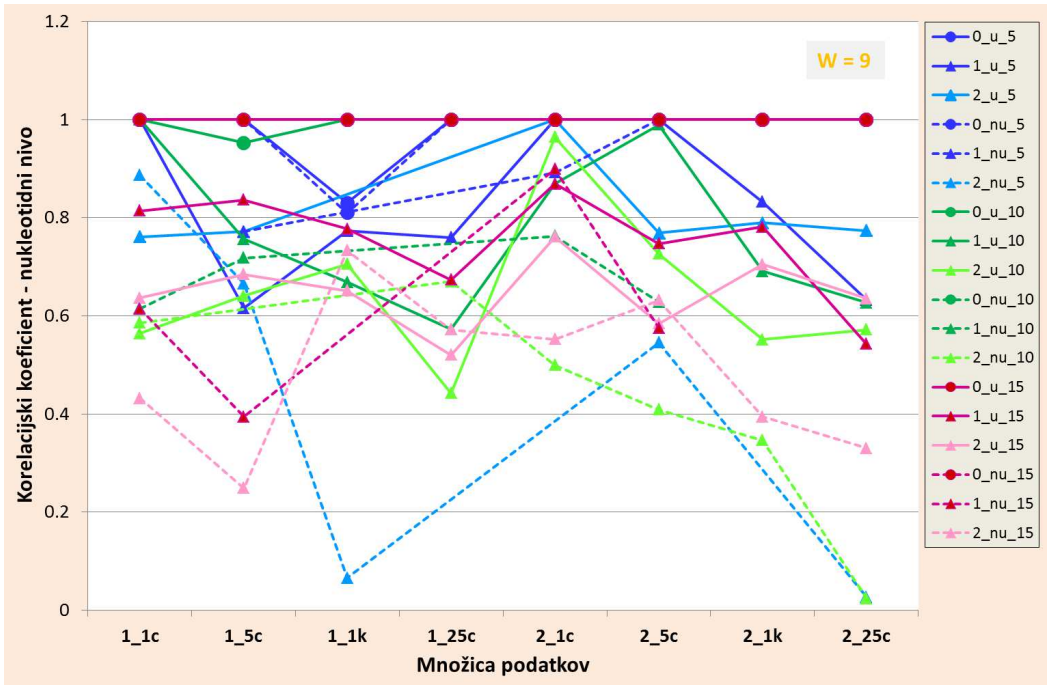

Slika C.21: Občutljivost algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 9$, vzorčni nivo.



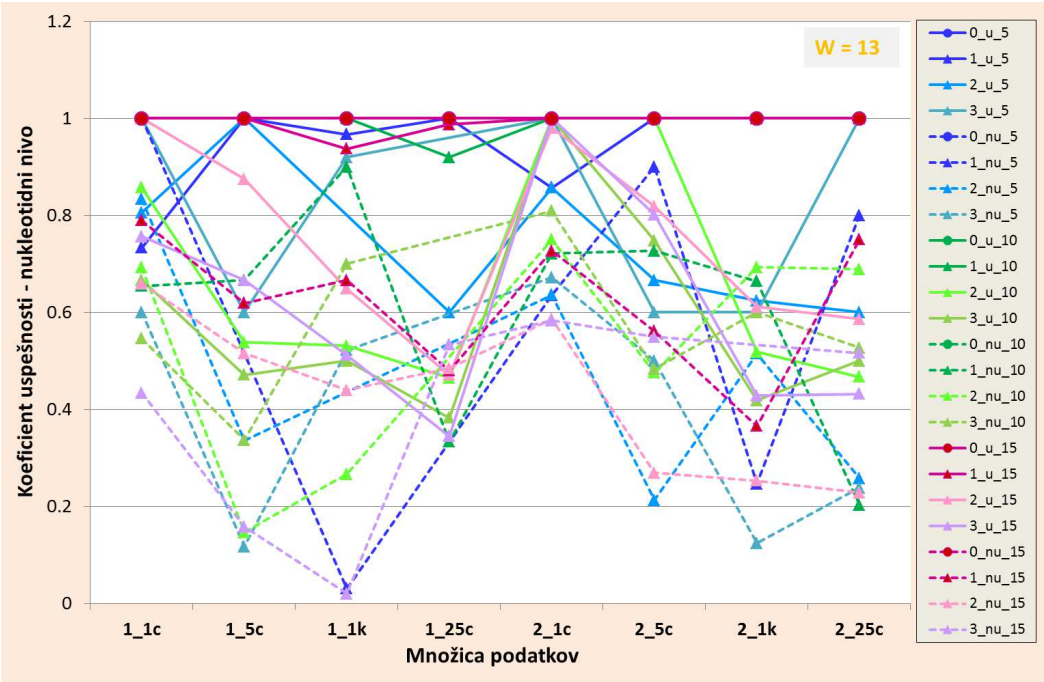

Slika C.22: Pozitivna napovedna vrednost algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 9$, vzorčni nivo.



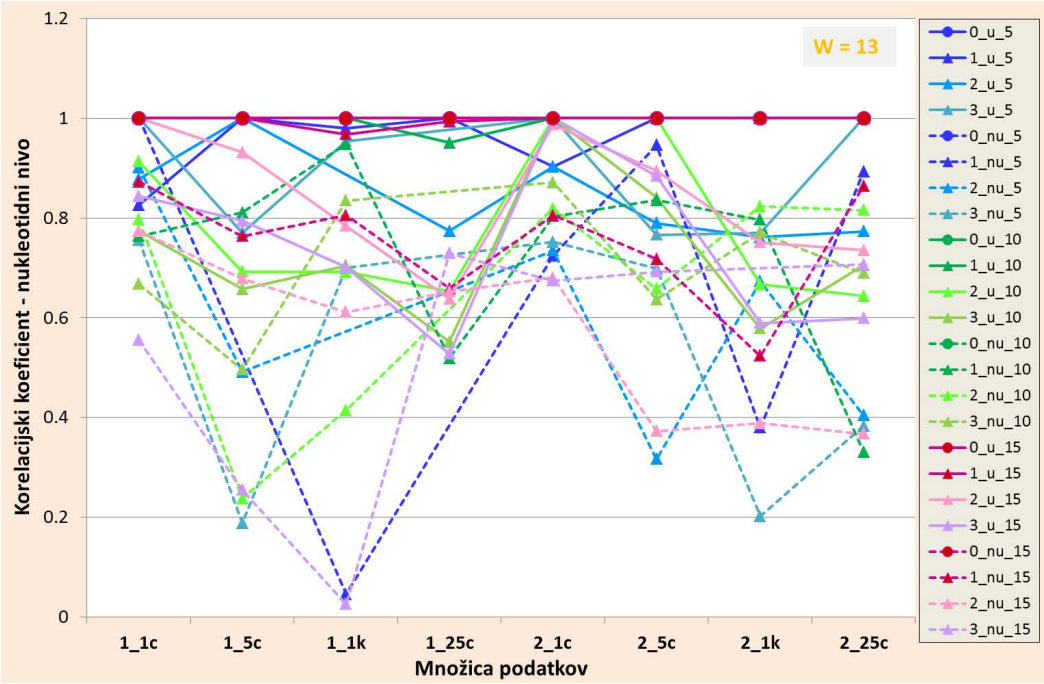

Slika C.23: Koeficient uspešnosti algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 9$, nukleotidni nivo.



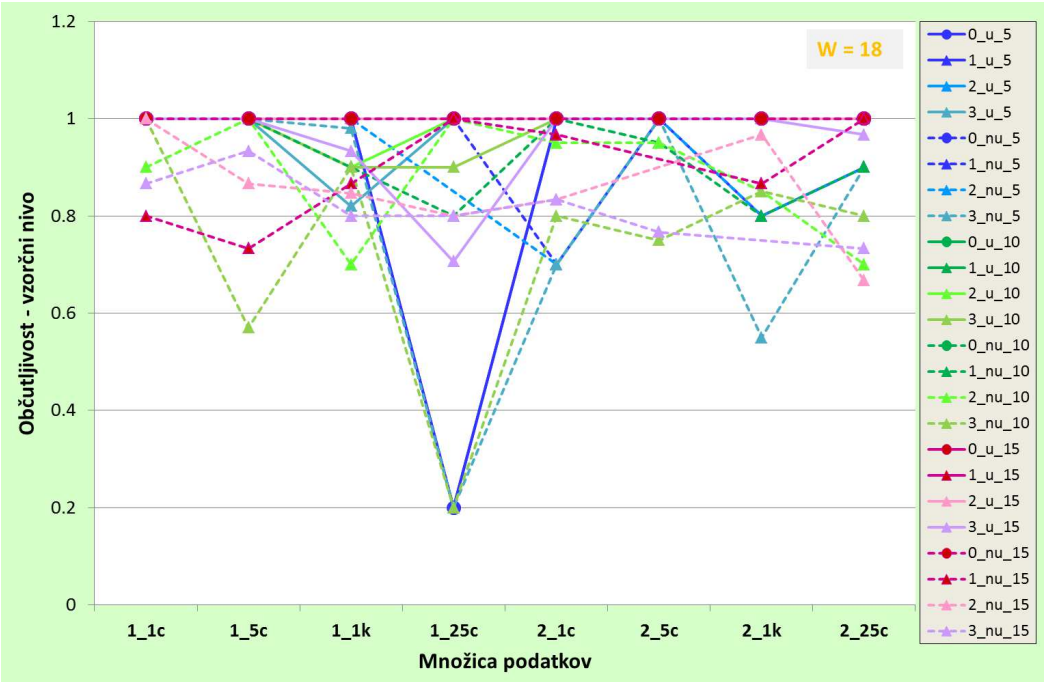

Slika C.24: Korelacijski koeficient algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 9$, nukleotidni nivo.



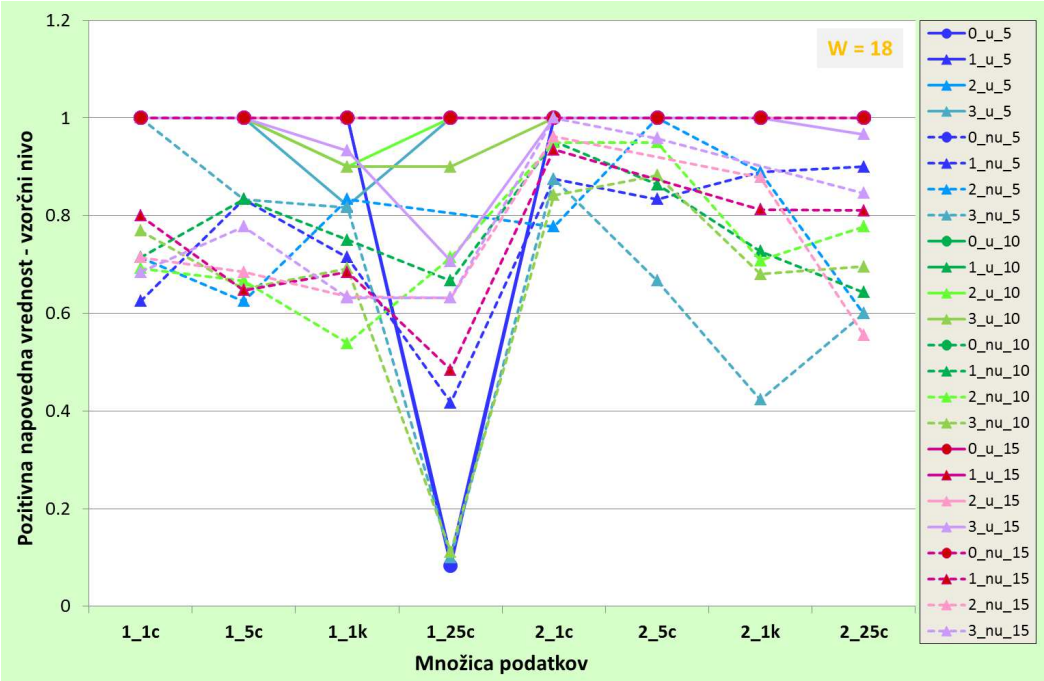

Slika C.25: Koeficient uspešnosti algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 13$, nukleotidni nivo.



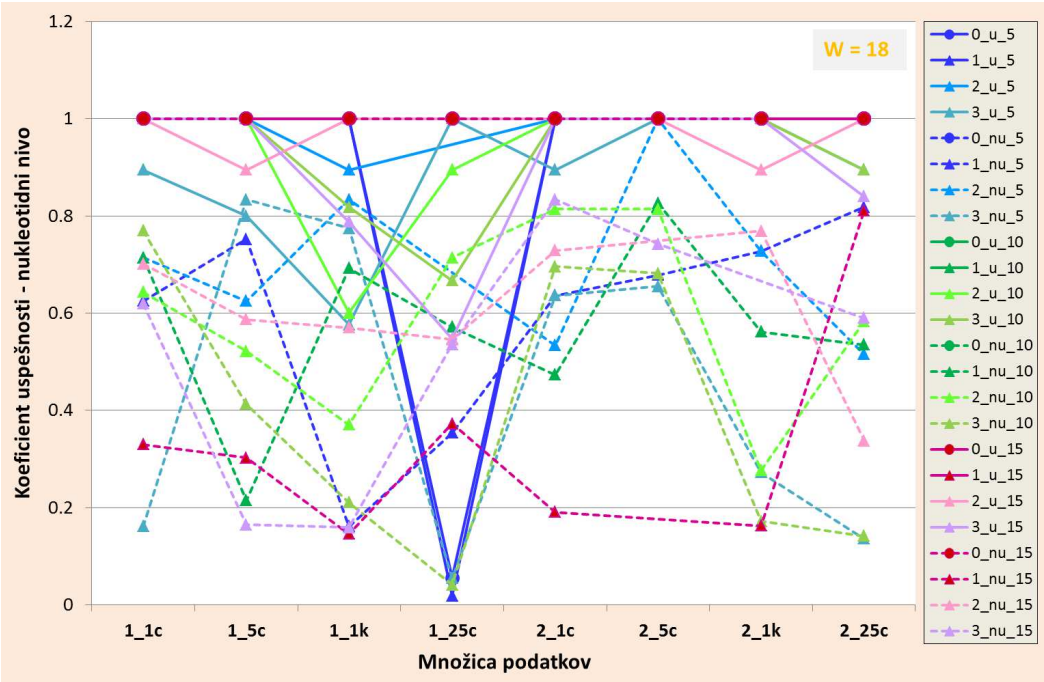

Slika C.26: Korelacijski koeficient algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 13$, nukleotidni nivo.



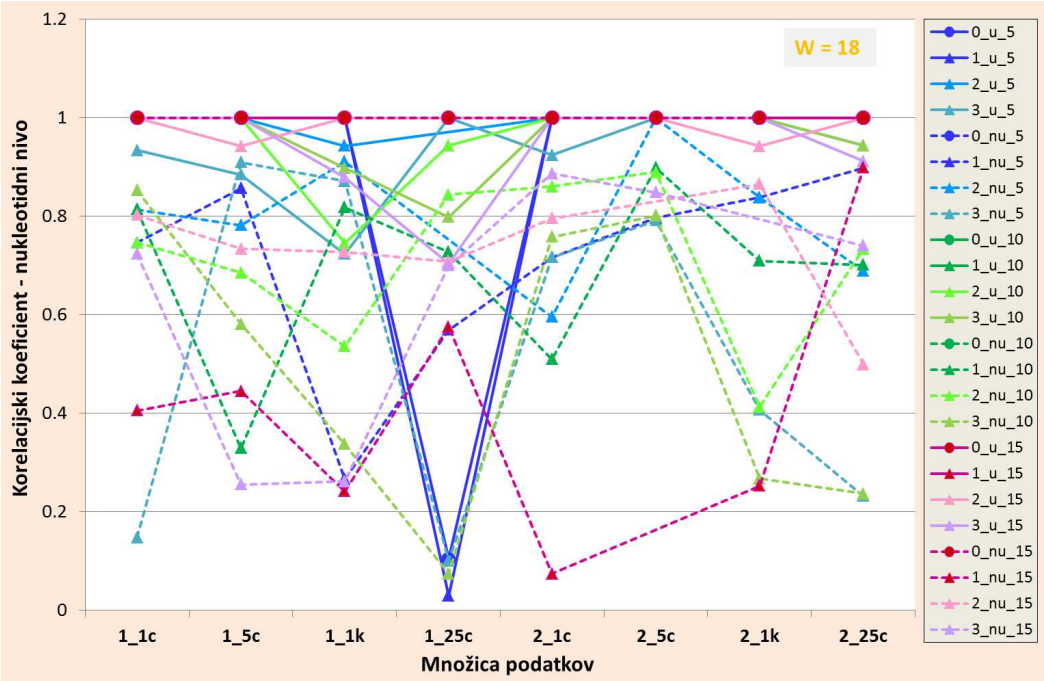

Slika C.27: Občutljivost algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 18$, vzorčni nivo.




Slika C.28: Pozitivna napovedna vrednost algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 18$, vzorčni nivo.




Slika C.29: Koeficient uspešnosti algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 18$, nukleotidni nivo.




Slika C.30: Korelacijski koeficient algoritma *GraphGibbs* na generiranih množicah z dolžino $W = 18$, nukleotidni nivo.

# Stvarno kazalo