Софийски университет „Св. Климент Охридски“
Факултет по математика и информатика

Изкуствен интелект в биоинформатиката: автоматизиран анализ и класификация на данни от паралелно секвениране

Милко Красномиров Крачунов

ДИСЕРТАЦИЯ

за присъждане на образователна и научна степен „доктор“ в професионално направление 4.6 „Информатика и компютърни науки“ научна специалност 01.01.12 „Информатика“ (Изкуствен интелект)

Научни ръководители:

проф. д-р Мария Нишева

доц. д-р Димитър Василев

София, 2014 г.

Съдържание

Съкращения 4

Речник на термините 5

1 Увод 13

1.1 Описание на проблема . . . 13

1.2 Значимост и актуалност на проблема . . . 15

1.3 Сложност на проблема . . . 17

1.4 Области и средства на изследването . . . 19

1.5 Цели и задачи на дисертацията . . . 21

2 Теоретични основи и състояние на изследванията по проблема 24

2.1 Теоретичен апарат на информатиката и изкуствения интелект . . . 25

2.1.1 Размити множества . . . 25

2.1.2 Изкуствени невронни мрежи . . . 26

2.1.3 Дърво на решенията и случайна гора . . . 35

2.2 Анализ на данни от паралелно секвениране . . . 37

2.2.1 Геномика . . . 37

2.2.2 Метагеномика . . . 39

2.3 Приложение на информатиката в геномните изследвания . . . 40

2.3.1 Алгоритми за съпоставяне . . . 40

2.3.2 Асемблиране . . . 46

2.3.3 Йерархично клъстериране и филогенетични дървета . . . 48

2.3.4 Откриване и корекция на грешки . . . 56

2.4 Съществуващи решения и алтернативни методи . . . 58
2.4.1 SHREC . . . 58
2.4.2 V-Phaser . . . 59
2.4.3 Coral . . . 61

3 Формализация и решение на проблема за анализ и класификация на данни от паралелно секвениране 63

3.1 Постановка на задачите . . . 63
3.2 Входни данни . . . 65
3.3 Предварителна обработка . . . 66
3.4 Разработени аналитични методи . . . 68
3.4.1 Аналитичен подход за корекция на грешки . . . 68
3.4.2 Индиректни подходи за валидация . . . 75
3.5 Размит индикатор на увереност в анализа на метагеномни данни . . . 82
3.6 Използване на невронни мрежи и други средства на изкуствения интелект 86
3.6.1 Вход . . . 87
3.6.2 Обучение с виртуални грешки . . . 90
3.6.3 Топология . . . 91
3.6.4 Дървета на решенията и други подходи . . . 91
3.6.5 Използване на получените модели . . . 91

4 Софтуерна реализация 93

4.1 Език за описание на управляеми работни потоци . . . 93
4.1.1 Описание на типовете данни . . . 95
4.1.2 Услуги, прототипи на методи и функции . . . 98
4.1.3 Управляеми работни потоци . . . 100
4.1.4 Отпратки към стойности и множества данни . . . 104
4.1.5 Език за изрази . . . 104
4.1.6 Деклариране на външни услуги . . . 106
4.2 Софтуерна система untangle . . . 109
4.2.1 Модули . . . 109
4.2.2 Инструменти за команден ред . . . 110

5 Анализ на резултатите от проведените експерименти 112

5.1 Изследване на разпределението на грешките . . . 113

5.2 Експериментални резултати за аналитичния подход . . . 114

5.2.1 Резултати от използване на сходството като тегло . . . 114

5.2.2 Резултати от прилагане на променлив праг . . . 117

5.2.3 Резултати от прилагане на острата мярка за сходство . . . 118

5.3 Приложение на невронни мрежи и други инструменти на ИИ . . . 119

5.3.1 Резултати от обучение на ИНМ и други модели на ИИ . . . 120

5.3.2 Приложение на методи на ИИ върху кандидати, избрани с аналитичния подход . . . 127

5.4 Сравнение със съществуващи решения . . . 133

5.5 Приложение на разработените методи при изследване на генома на пшеницата . . . 135

5.6 Оценка на потенциала на размития индикатор на увереност . . . 136

5.7 Обобщение на резултатите . . . 138

6 Приноси и перспективи 140

6.1 Теоретико-методични приноси на дисертационния труд . . . 140

6.2 Експериментални приноси на дисертационния труд . . . 141

6.3 Перспективи за бъдещо развитие . . . 142

6.4 Декларация за оригиналност . . . 143

6.5 Участия в проекти . . . 144

6.6 Публикации по темата на дисертационния труд . . . 144

6.7 Доклади . . . 145

6.8 Постери . . . 146

Литература 164

Списък на фигурите 165

Списък на таблиците 166

Съкращения

API application programming interface.

MRO method resolution order.

NGS next-generation sequencing.

SNP single nucleotide polymorphism.

база нуклеотидна база.

ИИ изкуствен интелект.

ИНМ изкуствена невронна мрежа.

ППИ приложно-програмен интерфейс.

СГ случайна гора.

Речник на термините

de novo секвениране

Секвениране на даден геном за пръв път.

k-мер (k-mer)

Подниз на символен низ, който има дължина k и се използва за ускоряване на процеса на търсене в изчислителната геномика.

аминокиселина (aminoacid)

Органично съединение, което е основна част от веригите на протеините. В изчислителната протеомика се обозначават с буква избрана от 20-буквена азбука, и секвенциите се подлагат на изследвания подобни на тези от геномиката.

ансамблов метод на обучение (ensemble method, ensemble learning)

Методи на машинното самообучение, които прилагат едновременно няколко различни модела на самообучение с цел постигане на по-добър резултат.

асемблиране (assembly)

Процес, при който къси прочити се съединяват, за да бъдат открити по-дълги последователности от секвенираните ДНК или РНК вериги.

булева мрежа (boolean network)

Модел, който представя експресията на гените като булеви променливи, чиито взаимозависимости са описани с булеви функции. Променливите са върхове в мрежа, чиято топология се определя от това между кои гени моделът предсказва пряка зависимост.

ген (gene)

Основен носител на генетична информация, представляваща група от нуклеотиди, която кодира протеин.

генна експресия (gene expression)

Процес на биологическо изразяване (проява) на информацията съдържана в даден ген, при който тя се преобразува в неговия краен продукт – най-често даден протеин.

генна регулаторна мрежа (gene regulatory network)

Слабо изследвано непряко биохимично взаимодействие между група гени, при което те взаимно регулират степента на генната си експресия, образувайки сложна мрежа за сигнализация. Често се моделира с помощта на различни математически модели (като булеви мрежи) или стохастични модели.

геномика (Genomics)

Дял от генетиката, който използва цифровите данни получени след процеса на секвениране и прилага методи на биоинформатиката за анализа им.

делеция (deletion)

Изтриване на единична база или аминокиселина, или последователност от бази или аминокиселини.

ДНК (DNA)

Дезоксирибонуклеинова киселина – макромолекула, която служи за съхранение и пренос на генетична информация.

дърво на решенията (decision tree)

Дърво, което спомага за взимането на решенията, като предлага прости правила за избор на път във всяко от разклоненията си. Използват се в изкуствения интелект, където съществуват алгоритми за самообучение строящи такива дървета.

еволюция (evolution)

Процес на развитие на биологичните организми, при който наследените от тях характеристики се променят постепенно с всяка генерация.

изчислително изследване (in silico research)

Научно изследване, което се извършва с помощта на компютър. Може да включва извършване на изчислителен анализ върху данни събрани от експеримент извършен в естествена среда, или извършване на компютърни симулации на естествена среда.

инсерция (insertion)

Вмъкване на единична база или аминокиселина, или последователност от бази или аминокиселини.

интервал (gap)

Незначещ символ вмъкнат в генетична последователност при процеса на съпоставяне, за да бъдат подравнени региони съвпадащи с региони в други генетични последователности. Интервалите съответстват на делеции в секвенцията, инсерции в останалите секвенции или на регион, който не е бил секвениран.

кеш (cache)

Кеш в информатиката означава структура от данни в компютърната памет, или физическа памет (кеш-памет), която съхранява избрани данни, за да бъдат те бързо достъпни при бъдещи запитвания.

контиг (contig)

Структура от припокриващи се ДНК фрагменти, получена при процеса на асемблиране, която представлява дадена по-дълга последователност от бази. Тя е междинен етап между кратките фрагменти и цялата генетична последователност. От *contig*uous (съседен, непрекъснат).

къс прочит / прочит (short read)

Кратка ДНК или РНК секвенция получена при процеса на секвениране. Обикновена има дължина от 35 до 500 бази.

масово паралелно секвениране (massive parallel sequencing)

Някой от множество подходи за високопроизводително ДНК секвениране, разчитащи на паралелизиране на процеса. Нарича се още секвениране от следващо поколение (next-generation sequencing, NGS) и секвениране от второ поколение (second-generation sequencing).

матрица на разстоянията (distance matrix)

Матрица, която включва разстоянията между двойките елементи в дадено множество. Много алгоритми за анализ на геномни данни разчитат на такава матрица, построена с помощта на разстояние на Хеминг между двойките секвенции.

метагеномика (metagenomics)

Дял от геномиката, който се занимава с изследване на биологични проби, съдържащи множество микроорганизми.

мода (в статистиката)

Най-често срещаната стойност в множество от наблюдения.

мононуклеотиден повтор (mononucletide repeat)

Последователност от една повтаряща се база (например AAAAA); кратък тандемен повтор от една база.

мутация (mutation)

Еволюционен процес, при който един или група нуклеотиди са били заменени с други.

нуклеотид (nucleotide)

Органична молекула, която участва във формирането на веригата на ДНК и РНК и образува техните последователности. Тя включва нуклеотидната база, която

носи генетичния му смисъл.

нуклеотидна база (nucleobase)

Органично съединение, което представлява най-малката единица генетична информация и е основен компонент на цифровите генетични данни, които подлежат на изследване с компютър. В ДНК може да бъде аденин (A), гуанин (G), цитозин (C) или тимин (T). В РНК тимина е заменен с урацил (U).

обучаващо множество (training set)

Множество от примери, които се използват за обучение на даден модел от машинното самообучение (за класификация или извършване на друг вид предсказания). При обучение с учител те съдържат както входните данни за модела, така и очаквания негов изход.

обучаващо правило (learning rule)

Обучаващо правило е метод за подобряване на качеството на предсказание на една изкуствена невронна мрежа, като теглата на мрежата се променят спрямо големината на грешката, която мрежата допуска на множество от обучаващи примери.

онтология (ontology)

В информатиката, средство за представяне на знания, което може да описва сложни взаимовръзки между понятията.

остра функция за сходство (sharp similarity function)

Термин, използван в този труд за функция на сходство между двойки символни низове, чиято стойност намалява рязко при натрупване на разлики между низовете. Тоест, функция имаща нелинейна („надлинейна“) зависимост от броя разлики.

повтор (repeat)

Последователност от бази, която се среща повече от веднъж на различни места в генома.

подрязване (trimming)

Премахване на излишните части от символен низ (например, интервали в краищата му). При анализа на данни секвенирани с платформата 454 означава премахване на опашките на секвенциите, в които се очаква да има високо ниво на шум.

прекомерно напасване (overfitting)

Един модел на статистиката или машинното самообучение има прекомерно напасване към данните, когато той описва шума и леките отклонения вместо търсената в данните зависимост. Това може да е причинено от използването на прекалено много параметри.

приложно-програмен интерфейс (application programming interface)

Интерфейс, предлаган от дадена операционна система, библиотека или програма поддържаща приставки, с който друг софтуер може да взаимодейства. Това може да включва комуникационни протоколи, абстрактни дефиниции на функции и методи и други.

РНК (RNA)

Рибонуклеинова киселина – макромолекула със сходни на ДНК свойства и функции, която е по-нестабилна, и в която нуклеиновата база тимин (T) е заменена с урацил (U).

самообучение без учител (unsupervised learning)

Метод за обучение на модел на ИИ, в който обучаващото множество не включва класификация на обучаващите примери, и обучаващия алгоритъм трябва да успее да я извлече от открити зависимости в данните.

самообучение с учител (supervised learning)

Метод за обучение на модел на ИИ, в който обучаващото множество включва класификация на обучаващите примери.

секвенатор (sequencer)

Апарат, който автоматизира процеса на разчитане на ДНК последователностите, записвайки ги в електронен вид като множество от символни низове, наречени кратки прочити.

секвениране (sequencing)

Процес на преобразуване на генетичните данни от биологична проба в електронен вид, подходящ за обработка с помощта на компютър.

секвенция (sequence)

Последователност от нуклеотиди или аминокиселини.

синтактичен анализатор (parser)

Синтактичните анализатори (или парсъри) са софтуерни инструменти, които обработват символен низ или друга поредица от знакове или думи според правилата на дадена формална граматика, генерирайки синтактично дърво или друга йерархична структура. Това подготвя данните за операции от по-високо ниво като компилиране и изпълнение на софтуер и други.

структурно вариране (structural variation)

Вариация между различни копия на дадена генетична последователност. Този термин обикновено се използва за вариациите, които не са точкови мутации и не представляват увреждане на хромозомата.

суфиксно дърво (suffix trie)

Дърво за бързо търсене в символен низ, в което пътищата от корена отговарят на ключове за търсене (за разлика от двоичните дървета, където ключовете са присвоени на възлите). Английският термин „trie“ идва от думата „re*trie*val“ и означава префиксно дърво.

съпоставяне (alignment)

Процес на обработка на данни, при който два или повече подобни символни низа с отклонения по дължина се сравняват, общите им региони се съпоставят и в

низовете се вмъкват интервали така, че общите части да заемат едни и същи позиции във всички низове.

тандемен повтор (tandem repeat)

Последователност от бази, която се среща повече от веднъж и срещанията ѝ са последователни. Кратките тандемни повтори (до 10 бази) се наричат микросателити, по-дългите минисателити.

тестово множество (testing set)

Множество от примери, отделно и независимо от обучаващото, което се използва за изпитание на вече обучен модел на машинно самообучение.

точкова мутация (single nucleotide polymorphism, SNP, /snɪp/)

Вариация между две генетични последователности, която представлява промяна в единствен нуклеотид. Може да бъде заместване, инсерция или делеция.

ϕенотип (phenotype)

Съвпкупността от видими характеристики, които представляват изражение на генотипа (съвкупността от гени), на даден биологичен вид. Те включват физиологически и биохимически характеристики, морфологични признаци, стадии на развитие и други.

филогенетичен (phylogenetic)

Обект на филогенията; свързан с изследване на еволюционните връзки между групи организми.

хипервариабилен участък (hypervariable region)

Участък от ДНК веригата, в който има високо ниво на вариация между близки видове.

Глава 1

Увод

Анализът на данни от паралелно секвениране в геномните изследвания е актуален проблем, който има значение за широк кръг от научни и практически области – от естествената история на живите организми до здравеопазването. В частност, метагеномиката, върху която е съсредоточен този труд, се занимава с изследване и класификация на големи групи микроорганизми, и със задачи като изучаване на биоразнообразието им, еволюционната им история и оценяване на ефекта им върху околната среда, добива на земеделска продукция и човешкото здраве.

Различните метагеномни изследвания са силно затруднени от липсата на добри решения за елиминация на шума в данните, предизвикан от грешки на апарата за секвениране. Това е най-видимо в задачите за автоматизирано откриване на структурни варианти, които разчитат на възможността откритите точки на вариация да могат да бъдат ефективно класифицирани като грешки или мутации. Информатиката предлага широк набор от инструменти, които имат потенциала да спомогнат за решаване на този проблем. По-специално, в този труд е разгледан резултатът от прилагането на мощните средства за класификация на изкуственият интелект при различаването на грешки от мутации.

1.1 Описание на проблема

Геномното секвениране (genome sequencing) е процес, при който генетична информация, съхранена биологически под формата на вериги от *нуклеотиди* (nucleotides), се

прочита от *секвенатор* и се прехвърля в паметта на компютърна система като последователности над четирибуквена азбука, съответстваща на четирите *нуклеотидни бази* (nuleobases), за да може да бъде обект на изчислителни изследвания [75]. *Масовото паралелно секвениране* или секвенирането от ново поколение (massively parallel sequencing, next-generation sequencing (NGS), second-generation sequencing) обхваща редица технологии за секвениране, които позволяват бърз прочит на много голям обем от кратки генетични фрагменти [3].

След извършване на процеса на секвениране, прочетените данни биват подложени на различни алгоритми за обработка преди те да станат обект на изследвания. Определени изчислителни задачи, като асемблирането на геноми на макроорганизми [164] и ДНК експертизите в съдебната медицина [64], стъпват на вече развити и утвърдени подходи. В същото време, все по-голяма популярност набират области като *метагеномиката* (metagenomics), занимаваща се с изследване на генетичен материал от групи микроорганизми, където задачите не винаги се решават оптимално от наличните средства [158, 61, 144]. Един от значимите проблеми, с които метагеномните изследвания се сблъскват, е голямото количество грешки при секвениране, при които погрешни бази са били заместени, изпуснати или добавени в прочетените последователности.

Проблемът за елиминация на ефекта на грешките в процеса на едно метагеномно изследване е комплексен. Налице е необходимост както от разработване на подходящи методи за оценката и откриването им, така и от разработване на подход, при който тази оценка да бъде включена в самото изследване – било то чрез подход за корекция на самите грешки или чрез модификация на използваните алгоритми за обработка и изследване. В настоящия труд са разгледани и двете възможности. Освен това поради спецификата на метагеномните данни, се налага и разработване на оригинален подход за валидация на предложените методи.

Поради необходимостта от запазване на редки различия в генетичните вериги като *точкови мутации* (single nucleotide polymorphisms, SNPs), които са важен обект на изследване, изследователите в областта са много консервативни към средствата за премахване на шум. В същото време, поради чувствителността към качеството на данните, всякакво подобрение е от много голямо значение за тях. Всеки разработван подход за справяне със зашумени данни трябва не само да покаже, че води до подобрение, но

и да покаже, че премахва възможно най-малко действителна вариация. Всякакъв напредък в качеството на класификация на откритите разлики като грешки или мутации (естествена вариация) е от голямо значение за този род изследвания.

Изкуственият интелект предлага средства, които са обещаващи при търсенето на решение на този проблем. Данните съдържат неструктурирани и хаотични връзки, които носят скрит биологичен смисъл, но не се подават на формализиране. Това предразполага към извършване на експерименти с апарата на изкуствените невронни мрежи [39], чийто потенциал се крие именно в способностите му да прави изводи точно на базата на такива неявни зависимости. От друга страна, апаратът на размитите множества [162] представя една възможност несигурните резултати, получени при оценка на грешките, да бъдат внедрени в алгоритмите за обработка, без да бъдат премахвани естествените вариации.

За потвърждение на качествата на разработените като част от този труд методи за справяне с грешките е необходимо да бъдат извършени комплексни изчислителни експерименти. В същото време самите процедури и работни потоци за изследване имат необходимост от гъвкавост, и подлежат на модификация за приложение на предложените тук средства. Това предразполага разработването на софтуерна система за описание на работни потоци, която позволява лесно и гъвкаво изпълнение на изчислителни експерименти.

1.2 Значимост и актуалност на проблема

Метагеномиката се занимава с изследване на местообитания и съобщества от микроорганизми, които имат значение за редица области. Изследването на съобществата, които обитават почвата [31] и растенията [25], е от ключово значение за разбирането на част от процесите, които диктуват оцеляването и развитието на културите в земеделието [77]. Изследването на съобществата, които обитават човешкия организъм, е ключово за разбиране на значението им за човешкото здраве [105]. Изследването на вирусните метагеноми [73] е важно за изследването на еволюционните процеси, които предизвикват бъдещи епидемии, както и при създаването на лекарства, които ги предотвратяват.

Грешките – представляващи подменени, изпуснати и вмъкнати от секвенатора бази – са сериозен проблем за много от решаваните задачи. При изследване на биоразнообразието на съобществата от микроорганизми в дадена среда, грешките при секвениране могат да доведат до значително завишени предсказания [74, 62] или да променят значително структурата на реконструираното *филогенетично* (еволюционно) дърво [117]. При търсене на редки точкови мутации, наличието на грешки води до значително повишаване на погрешните предсказания [23]. Честа практика при секвениране е да се отстраняват *прочити* (reads), в които има съмнения за грешки [63], както и прочити, които се срещат само по веднъж [144].

Наличните подходи, разгледани в 2.3.4 и 2.4, не успяват ефективно да разрешат проблема с шума в данните. Създаването на усъвършенствани средства за справяне с грешките в данните може да спомогне за подобряването на изследванията в различни ключови области по редица начини.

- Подобряване на качеството ще помогне за редуцирането на количеството отстранени данни, което ще увеличи количеството полезна информация, която може да се изведе от една проба, и ще намали цената на процеса на секвениране.
- Намаляване на броя грешки и количеството загубена информация в следствие на отстраняване на прочити ще спомогне за увеличаване на броя на открити редки точкови мутации.
- Дори малък принос за подобрение на средствата за справяне с грешките дава основа за бъдещ труд, който е необходим, за да се елиминират напълно отклоненията при оценяване на биоразнообразието, както и несъответствията и неточностите при откриване на точкови мутации.
- Промяна на начина на използване на оценките за качество може да отложи отстраняването на данните, считани за погрешни, или да ги остави достъпни за по-късен анализ.
- Промяна на алгоритмите за обработка, като разгледаните в 2.3, така че да вземат предвид предполагаемите грешки, може да позволи селективен избор на местата, където да се прилага премахване на шум – стъпките, които са силно уязвими

при наличие на грешки, могат да бъдат настроени да заобикалят зашумените участъци, докато стъпките, които разчитат на запазени вариации, могат да работят с непроменените данни.

Проблемите, с които се занимава този труд, имат значителни сходства с проблеми извън областта метагеномиката. Като пример може да се разгледа изследването на генома на пшеницата. От една страна, като основна култура, използвана за изхранване на населението, изследването му има изключителна важност. От друга, изключително големият размер на генома на вида, както и структурата му – включваща три сходни подгенома – сериозно затруднява извършването на всякакъв вид анализ. До този момент секвенирането на пълния геном на пшеницата, чрез извършване на успешна процедура по асемблиране като описаната в 2.3.2, е било неуспешно [21].

Затрудненията при работата с пшеницата, свързани с откриване на редките вариации, с различаването на грешки от точкови мутации, както и като цяло с работата със сходни последователности, силно напомнят за хетерогенната структура на данните, които се срещат в едно метагеномно изследване. Това предоставя възможност представената тук работа да бъде адаптирана в бъдещ труд за проблеми като тези при пшеницата, както и да спомогне за разрешаването на по-голям клас задачи от пряко поставените.

В същото време, наличните софтуерни средства за интеграция на изчислителни инструменти в областта на геномиката и биоинформатиката имат редица недостатъци. Налага се изследователите да интегрират отделните инструменти ръчно, както и често да извършват едни и същи технически процедури независимо, поради липсата на добри средства за споделяне. При разширението на разработвания към този труд софтуер до една пълна среда за описание и автоматизирано изпълнение на работни потоци за геномни и биоинформатични задачи, той би създал още една възможност за улесняване на изследователите в областта.

1.3 Сложност на проблема

Метагеномните проби съдържат множество микроорганизми, и по тази причина след клъстериране отделните групи в секвенираните данни са хетерогенни и съдържат

сходни последователности от множество геноми, за разлика от секвенирането на отделни видове, където пробите съдържат прочити на един и същ отрязък от ДНК веригата на даден геном. Това затруднява откриването на грешки, допуснати при процеса на секвениране – те включват подменени, вмъкнати и изпуснати от секвенатора бази, които съответстват на съществуващата естествена вариация.

Разработката на средства за справяне с грешките в метагеномни данни от паралелно секвениране може да срещне трудности, които могат да бъдат разделени в няколко категории.

- Хетерогенната структура на данните.
- Очакванията на изследователите за запазване на естественото вариране и други характеристики на данните.
- Трудностите при валидация и сравнение на различните подходи.
- Липсата на добри обучаващи множества за прилагане на методи на машинното самообучение.

Хетерогенна структура на данните. За разлика от обичайното геномно множество от данни, където присъстват само последователностите на един организъм и сходни региони се срещат рядко, една метагеномна проба се състои от множество различни сходни микроорганизми, голяма част от които – неизвестни. Това значително усложнява процеса на различаване на сигнал (естествени вариации) от шум (грешки), особено в редките видове.

Запазване на естественото вариране. Допълнително усложнение представляват и точковите мутации, които от една страна предизвикват широк интерес у изследователите, а от друга страна силно приличат на грешките – и двете представляват случайни единични *заместванията*, *инсерции* и *делеции*[1], които могат да участват само в малък брой от видовете в пробата. За изграждане на добър подход за откриване на грешките, различаването им е наложително, въпреки значителната сложност на тази задача.

[1] В биологията „инсерция“ и „делеция“ са се наложили като превод на „insertion“ и „deletion“, и означават вмъкване и изтриване на бази в ДНК верига

Валидация и обучаващи множества. За разлика от секвенирането на един единствен биологичен вид, всяка метагеномна проба съдържа значително и специфично за нея биоразнообразие, като няма две метагеномни проби, които да са еднакви. Всяка се характеризира със специфично разпределение на видовете в нея, или с присъствието на определени редки видове. Често пъти точно тези характеристики представляват обект на изследване. Това означава, че повторното секвениране на дадена проба не е възможно, което от своя страна прави невъзможно пълното възстановяване на реалните данни в нея.

Проблем е и това, че самите проби представляват съвкупности от известни и множество неизвестни видове, чието пълно откриване и класификация е сред основните задачи на метагеномиката. Тези видове не могат да бъдат ефективно отделени в лабораторни условия, нито да бъдат култивирани извън естествената им среда, поради което задачата трябва да се решава изчислително.

Процесът на валидация е силно затруднен, тъй като не е възможно да бъде извършен директно върху сигурна референтна проба. По същия начин е възпрепятствано и създаването създаването на добри обучаващи и тестови множества за методи на изкуствения интелект.

В този труд са предложени два индиректни подхода за валидация, които са приложими върху метагеномни данни, заедно с индиректен подход за обучение на невронни мрежи и различни класификационни методи от машинното самообучение.

1.4 Области и средства на изследването

Биоинформатика. *Биоинформатиката* е интердисциплинарна научна област, която се занимава с изследването на данни от различни сфери на биологията със средствата на информатиката. Тя силно разчита на разработване на алгоритми и софтуер, които да могат да бъдат използвани както за обработка на постъпващата информация, така и за извършване на числови експерименти върху нея.

Анализът на генетични последователности намира своето приложение от изучаването на генетични увреждания до съдебната медицина. Изграждането на модели на *генната експресия* и на *генните регулаторни мрежи* с помощта на булеви мрежи на-

мира своето приложение в диагностиката на заболявания като търси връзките между проявата на група гени и проявата на тези заболявания. *Генетичните* и *фенотипните онтологии* помагат за частично описание на сложните зависимости, които съществуват в един биологичен организъм и обвързват генетичните му характеристики, анатомичните му признаци, морфологични му признаци, стадиите на растежа му и поведението му. Изграждането на *модели на еволюцията* и строенето на *филогенетични (еволюционни) дървета* спомага за разбирането на спецификите на процеса, както и разкрива какви исторически са роднинските връзки между видовете. Предсказването и изследването на *протеинови структури* намират своето приложение при дизайна на нови лекарства.

С развитието на информационните технологии, както и с развитието на технологиите в областта на биоинформатиката (като например оборудване за ДНК секвениране) обемът данни, налични за обработка, нараства с много бързи темпове, понякога по-бързо и от самите изчислителни ресурси, с което силно нарастват и възможностите за разширяване на изследванията и постигане на нови, много по-значими резултати във всяка една насока, докато в същото време средствата на биоинформатиката стават все по-достъпни. Ако преди години цената на едно ДНК изследване е била непосилна, днес ДНК анализът е нормална практика при криминалните разследвания. По същия начин нарастват и темповете, с които нови лекарства се разработват с помощта на изчислителни средства.

Изкуствен интелект. *Изкуственият интелект* е област от информатиката, която се занимава с изграждане на интелигентни софтуерни системи, които могат да решават проблеми, обикновено считани в компетенцията на хората, с изчислителни средства. Те могат да включват всякакви задачи, които не се подават на решаване с формално описани алгоритми – разпознаване на образи, разпознаване на реч, роботика, самообучаващи се програми, изчислително творчество и други.

Решаваните проблеми са или прекалено сложни за изчисление, или нестрого определени, могат да налагат вземане на решения при частично налична информация, или при наличие на много несвързана информация, която не може да бъде филтрирана. Решенията разчитат на подходи, които включват голяма степен на несигурност,

стъпват на емпирични резултати и често пъти са недостъпни във вид, който може да бъде разбран и анализиран.

Изкуственият интелект е област, която се развива изключително бързо, успоредно с увеличаващите се изчислителни способности на модерните компютърни системи. Все повече ключови задачи намират своето решение. Докато разпознаването на печатни текстове отдавна е практически решен проблем, за който от много време насам не се налага използването на човешки труд, в последните години по-трудни задачи започват да намират своето решение. Системи, които разпознават глас, започват да стават все по-използвани, качеството на компютърния превод достига до ниво, при което се превръща в практически полезно помощно средство, разширява се използването на медицински роботи, а автоматизацията все повече навлиза и в други области, като например транспорта под формата на автономни автомобили.

1.5 Цели и задачи на дисертацията

Основната цел на този труд е да се разработят и изследват нови подходи за откриване и коригиране на грешки в метагеномни данни от паралелно секвениране, използвайки средствата на информатиката и изкуствения интелект.

Аналитичен подход за оценка на грешките. Основна задача в този труд е разработването на аналитичен подход за откриване на грешките, който разчита на ясно формулиран метод, основаван на локалните сходства между последователностите, с помощта на които се групират близките секвенции, използвайки тегла. Тя може да се раздели на следните конкретни задачи:

1. Да се предложи подход за използване на локалното сходство за оценяване на грешките.
2. Този подход да се включи във функция, която генерира числова оценка в интервала $[0, 1]$, която участва в класификатор, основан на числови прагове.
3. Да се разработят и изследват различни мерки за оценка на локалното сходство.

Приложение на оценките на грешките. Има повече от един начин една числова оценка да участва като инструмент за справяне с грешките. Естеството на данните води до очаквана несигурност в оценките, което обосновава изследването на повече от една алтернатива, и в частност изпълнението на следните задачи:

1. Да бъде предложен подход за корекция на грешките, основан на изчислените оценки и класификатора, използващ числови прагове.

2. Да бъде разработена алтернатива, при която оценките се прилагат направо в алгоритмите за обработка на генетичните данни като размит индикатор на увереност чрез въвеждане на претеглени функции за разстояние.

Валидация. Важно условие, за да може да бъдат оценени и сравнени методите за оценка и корекция на грешките, е да бъдат налични средства за валидация. Използването на преки подходи за валидация е затруднено от липсата на референтни множества от данни, което води до разширяване на списъка със задачи:

1. Да се разгледа възможността за симулиране на грешки за целите на валидацията.

2. Да се предложат различни подходи за индиректна валидация.

3. Да се сравнят всички предложени методи за откриване и корекция с помощта на предложените методи за валидация.

Оценяване на грешките с невронни мрежи. Апаратът на невронните мрежи дава много добра алтернатива за решаване на такъв вид проблеми. Оценяването и приложението му за класификация на търсените грешки включва изпълнението на следните задачи:

1. Да се предложи индиректен подход за обучение на невронната мрежа, основан на симулация на грешки.

2. Да се построи и обучи невронна мрежа на негова база, която да се използва за класификация на нуклеотидните бази на погрешно и правилно прочетени.

3. Да се комбинира резултатът, получен от аналитичния подход, с резултата, получен от невронната мрежа.

4. Да се оцени резултатът от прилагането на невронната мрежа в сравнение с резултата, получен само с аналитичния подход.

5. Да се използват и други средства за машинно самообучение, като например дървета на решенията, и резултатът да се сравнят с този от невронната мрежа.

Софтуер. Да бъде разработена гъвкава софтуерна система за автоматизирано изпълнение на разгледаните в труда процеси, чийто дизайн да позволява лесно разширяване към други цели и задачи.

Глава 2

Теоретични основи и състояние на изследванията по проблема

Навлизането на масово паралелно секвениране в метагеномиката води до засилено изследване на биоразнообразието на микроорганизмите, обитаващи различни среди, както и еволюционната им класификация. Анализът на данните при тези изследвания е силно затруднен от наличието на грешки и практическото отсъствие на добри средства за справянето с тях. Предвид значението им за широк набор от проблеми, включително добива на земеделска продукция и човешкото здраве, качеството на получените резултати има съществена значимост. Макар и в по-малка степен, същият проблем съществува и при изследването на някои определени растителни видове като пшеницата.

Информатиката и изкуственият интелект предлагат инструменти, които имат значителен потенциал да бъдат приложени към задачата за подобряване на качеството. Относителната комплексност на данните и отсъствието на директни методи за изчистване на грешките подчертано обосновават използване на апарата на изкуствения интелект.

В тази глава е направен както обзор на средствата на информатиката, които ще бъдат приложени, така и подробно представяне на този вид метагеномни изследвания, и наличните средства за справяне с грешките.

2.1 Теоретичен апарат на информатиката и изкуствения интелект

2.1.1 Размити множества

Изпълнението на представените подходи за откриване на грешки води до изчисляване на размити оценки за степента на увереност. Прекият начин тези оценки да се приложат е като се изчисли праг на грешката, който най-добре разделя оценените данни на верни и погрешни. Прякото приложение може да бъде разширено, като оценките бъдат приложени директно в алгоритмите за обработка на данните от паралелно секвениране, използвайки апарата на размитите множества [162] и размитата логика [109]. Размити множества, и в частност интуиционистки размити множества, са успешно прилагани при сходната задача за откриване на точкови мутации [97]. В тази работа е използван малко по-различен подход за тяхното приложение, за да се вземат предвид потенциалните грешки.

Размитото множество (fuzzy set) A описва обобщено множество, чиито елементи имат степен на принадлежност към това множество, дефинирана с помощта на функция на принадлежност, приемаща стойности в интервала $[0, 1]$. В този труд, степента на сходство на елемента x към размитото множество A ще се бележи с $A(x)$. Функцията на принадлежност $A(x)$ също е обобщение на индикаторната функция на множество.

Всички операции върху множество, като допълнение c, обединение $\cup$ и сечение $\cap$, имат разширения за размити множества, и следният набор от операции се използват най-често:

$$cA(x) = 1 - A(x) \tag{2.1}$$

$$(A \cup B)(x) = \max(A(x), B(x)) \tag{2.2}$$

$$(A \cap B)(x) = \min(A(x), B(x)) \tag{2.3}$$

Разглежданите алгоритми и тяхната софтуерна реализация позволяват използване на произволно избран набор от операции, които удовлетворяват следните групи от аксиоми за съответните операции върху размити множества.

Дефиниция 1 (Размити допълнения). Функцията c определя допълнението cA на размитото множество A, ако са в сила следните условия: *а*) c е непрекъсната и монотонна функция в $[0,1] \to [0,1]$; *б*) c изпълнява граничните условия $c(0) = 0$, $c(1) = 1$; *в*) c е своята обратна функция, $c(c(x)) = x$ за всяко x.

Дефиниция 2 (Размити сечения). Функцията I определя сечението $A \cap B = I(A, B)$ на размитите множества A и B, ако са в сила: *а*) I е непрекъсната, монотонна, асоциативна и комутативна функция в $[0,1] \times [0,1] \to [0,1]$; *б*) I изпълнява граничното условие $I(x,1) = I(1,x) = x$; *в*) $I(x,x) \leq x$.

Дефиниция 3 (Размити обединения). Функцията U определя обединение $A \cup B = U(A, B)$ на размитите множества A и B, ако са в сила: *а*) U е непрекъсната, монотонна, асоциативна и комутативна функция в $[0,1] \times [0,1] \to [0,1]$; *б*) U изпълнява граничното условие $U(x,0) = U(0,x) = x$; *в*) $U(x,x) \geq x$; *г*) когато $x_1 < x_2$ и $y_1 < y_2$, то е изпълнено стриктното неравенство за монотонност $U(x_1, y_1) < U(x_2, y_2)$.

За простота там, където се налага дефиниране на размити множества чрез прекалено сложни конструкции, същите означения в текста на настоящия труд ще се използват и за размита логика. Ще се приема, че при прилагането на логически операции върху функциите на принадлежност, резултатът е същият като на еквивалентната операция между множества. Така например, за стандартните операции ще бъде в сила следното:

$$\neg A(x) = cA(x) \tag{2.4}$$

$$A(x) \vee B(x) = (A \cup B)(x) \tag{2.5}$$

$$A(x) \wedge B(x) = (A \cap B)(x) \tag{2.6}$$

Така дефинирани, отрицанието $\neg$, дизюнкцията $\vee$ и конюнкцията $\wedge$ отговарят на безкрайнозначна логика на Лукашевич и t-норма на Лукашевич [85, 109].

2.1.2 Изкуствени невронни мрежи

Изкуствените невронни мрежи (ИНМ) са инструмент на изкуствения интелект, взаимстван от невронните мрежи, които стоят в основата на човешките мозък и нервна систе-

ма. Те подлежат на самообучение и са най-често използваният инструмент, прилаган в разпознаването на образи и други сходни задачи като разпознаването на реч.

2.1.2.1 Първоизточник на ИНМ – човешката нервна система

В една органична нервна система информацията се предава чрез електрохимичен процес, в който невроните играят ключовата роля да приемат, изпращат и обработват информация. Отделните неврони имат къси израстъци, наречени дендрити, през които те приемат електрическите сигнали, и един дълъг израстък наречен аксон, през който те ги изпращат. Клетъчното им тяло е електрически възбудимо и когато събраният сигнал, идващ през дендритите, достигне определен праг, невронът се активира и се стига до значително повишаване на електричния му потенциал. Потенциалът достига до аксона и от него се предава до дендритите на следващите неврони [8].

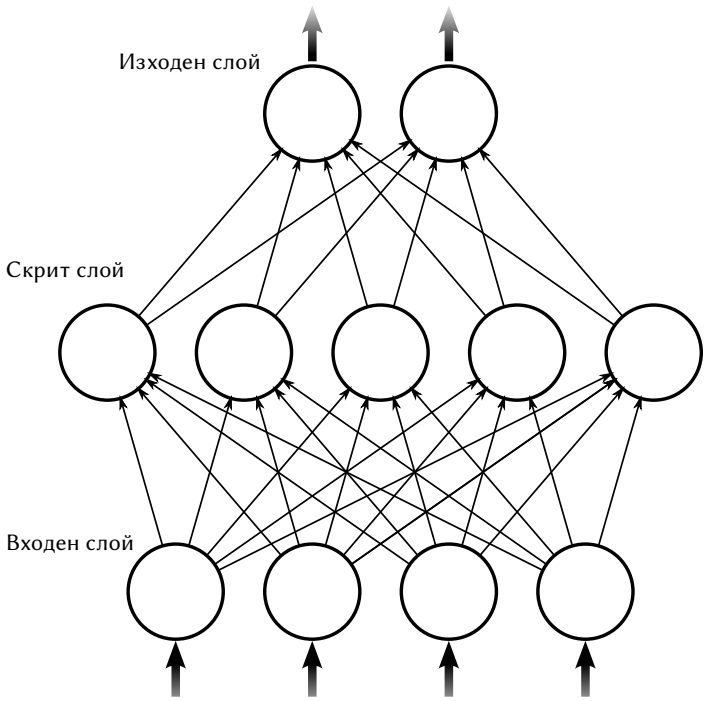
Връзката между аксона и дендритите се осъществява от синапси. Синапсите свързват нервните клетки с различни други клетки, като тази връзка може да се извърши чрез отделяне на химически сигнални вещества или чрез пряко протичане на електрически ток, което е възможно чрез свободен поток на йони между клетките. Химическите сигнални вещества могат да инхибират действието на следващите неврони, както и да имат значително комплексно действие. За сметка на това електрическите синапси позволяват много по-бързо предаване на сигналите между отделните неврони [8, 155].

Синапсите притежават свойството пластичност. То им позволява с течение на времето тяхното действие да отслабва или да се засилва и е в основата на човешката памет и способност за учене. Промяната в действието им зависи от реакцията на останалите клетки. Например, открито е наличието на силна връзка между тази промяна и отделянето на калций от клетката, която стои на изхода им [46]. Съществуват и електрически процеси, при които изходният неврон може да изпрати сигнал обратно във входния като реакция на пристигащ сигнал [140], като това също би могло да доведе до трайни промени в действието му.

За разлика от алгоритмите за обучение на ИНМ, разгледани по-долу, нито един от тези процеси сам по себе си не може да обясни напълно обучението при наличие на много нива от неврони, поради сложността на процесите в нервната система и мо-

зъка. За сметка на това, съществуват успешно прилагани изчислителни алгоритми за самообучение на изкуствени невронни мрежи, които почиват изцяло на известните биологични процеси, илюстриращи как такова обучение в действителност е възможно [17].

2.1.2.2 От естествени към изкуствени невронни мрежи



Фигура 2.1: Изкуствена невронна мрежа

Изкуствените невронни мрежи, илюстрирани на фигура 2.1, използват опростен модел на описаното дотук взаимодействие. Всяка ИНМ представлява мрежа от възли, наречени неврони. Всеки неврон притежава своя активационна функция, която съответства на прага на активация на естествените неврони. Входът им се пресмята чрез линейна комбинация, която е аналог на комбинацията от възбуждащи и подтискащи синапси в естествените невронни мрежи. Невроните се обособяват в слоеве (един входен, един изходен и нула или повече скрити – виж 2.1.2.3), като сигналът се предава от всеки слой към следващия, макар и да съществуват и по-сложни модели като рекурентните ИНМ, където подобно обособяване не може да се направи или остава само условно.

Обучение на ИНМ се извършва с помощта на *обучаващо правило* (learning rule), което се използва за промяна на теглата на връзките между невроните в мрежата, тоест на коефициентите в линейните комбинации. Това правило наподобява процесите на реакция в естествените невронни мрежи, и често се извършва с алгоритъма за обратно разпространение на грешката (backpropagation) [152, 153], разгледан по-подробно в 2.1.2.5.

За първи път през 1943 г. Маккълох и Питс създават изчислителен модел, взаимстван от нервната система, който те наричат прагова логика (threshold logic) [91]. По същото време Доналд Хеб изгражда теоретичен модел на пластичността на невроните и тяхната способност за самообучение [53]. На негова основа веднага бива разработен и изчислителен модел симулиращ самообучаващи се системи с помощта на компютър [40].

Перцептронът е първата реализация на невронна мрежа, която има вида на модерните ИНМ. Той е разработен като вероятностен модел, имащ за цел да подобри разбирането на човешките възприятия и мислене, и в първоначалния си вид представлява невронна мрежа без скрити слоеве, която понякога се нарича еднослойна. В нея входовете отговарят на стимули (като сигнали, получени от ретината) и са свързани направо с всеки изход с помощта на единствена линейна комбинация, която съответства на способността на човека да прави асоциации. Така изходите – представящи различни въздействия на стимулите вътре в организма – са представени от един линеен модел за класификация [125].

Така построен, перцептронът има изключително ограничени възможности за разпознаване на образи, като дори не е способен да моделира изключващо „или“ (xor) [96], поради което повечето изследвания се насочват към многослойни разширения. Между входния и изходния слой се въвежда слой от скрити неврони, който позволява моделирането на по-сложни функции. В същото време започва и разработването на по-сложни алгоритми за обучение, като алгоритъма за обратно разпространение на грешката [152, 153].

2.1.2.3 Характеристики на невронните мрежи

В една изкуствена невронна мрежа всеки неврон получава сигналите от невроните, с които е свързан, и ги обобщава, като прилага тяхна линейна комбинация с адаптивни тегла, чиито стойности се избират на базата на обучаващо правило. След като се пресметне линейната комбинация, върху резултатът се прилага активационна функция, която определя изходния сигнал на неврона. Тази функция най-често е или стъпаловидна функция, приемаща стойности измежду $\{0, 1\}$, или сигмоидна функция, приемаща стойности в $(-1, 1)$ [122].

При правите мрежи невроните се делят на три групи:

1. Входни неврони. Всеки входен неврон изпраща като сигнал една числова стойност от входните данни.

2. Скрити неврони. Сигналите, изпращани от скритите неврони, се изчисляват като функция на стойностите на сигналите на входните неврони, или на скритите неврони от предишния слой, ако в мрежата има повече от един скрит слой.

3. Изходни неврони. От изходните неврони се получава крайният резултат от приложение на мрежата. Стойностите на изходните им сигнали се изчисляват като функция на стойностите, получени от невроните на последния скрит, слой или от входните неврони при мрежи без скрити слоеве.

Всеки неврон y е свързан с невроните x от предходния слой. Неговият сигнал се изчислява като функция K от линейна комбинация на сигналите на невроните от предишния слой.

$$y_j = K\left(\sum_i w_{ij}^y x_i\right) \tag{2.7}$$

K се нарича активационна функция.

2.1.2.4 Класификация на невронните мрежи

Невронните мрежи могат да бъдат класифицирани по няколко начина.

На прави и рекурентни. Най-често използвани са правите ИНМ, при които сигналът се предава от един слой неврони към следващия. Те се наричат още ациклични, понеже представляват ацикличен граф. Съществуват и така наречените рекурентни ИНМ [151], при които обособяването в слоеве губи своя смисъл, понеже връзките могат да образуват цикли. Такива мрежи имат естествено вътрешно състояние, което им позволява да работят със последователности от данни, или данни разпределени във времето. Пример за проблем, при който съществуват такива данни, е задачата за разпознаване на реч, където рекурентните ИНМ намират своето приложение [50]. Тъй като значението на гените също се кодира в последователности, в които реда на нуклеотидните бази има значение, ацикличните мрежи са също и потенциален инструмент за задачите свързани с анализ на данни от геномно секвениране.

На еднослойни и многослойни. Еднослойните[1] мрежи, или мрежите без скрити слоеве, имат само входен и изходен слой, докато многослойните могат да имат един или повече скрити слоя. Броя на невроните, както и на скритите слоеве обикновено се определя емпирично, макар че изборът стъпва на основата на натрупания опит, свързан с решаването на подобни задачи, както и на други рационални съображения.

Според активационната функция. Могат да се използват различни активационни функции. За задачи, които включват работа само с булеви стойности, популярни са стъпаловидните функции, като:

$$K(s) = \begin{cases} 0, & s < 1/2 \\ 1, & s \geq 1/2 \end{cases} \tag{2.8}$$

За задачи, които включват работа с непрекъснати стойности, трябва да се използва алтернатива на стъпаловидна функция, която е непрекъсната и диференцируема. Популярни са различни сигмоидни активационни функции, които приемат непрекъснати стойности в някакъв интервал, най-често $(-1, 1)$. Сред най-често използваните е

[1] Различните автори използват различен начин за броене на слоевете. Някои броят всички слоеве с неврони, други само скритите слоеве, а трети само слоевете с неврони, чиито стойности са резултат от изчисление. Макар и в този труд да е възприета последната терминология, навсякъде топологията е била указана изрично.

логистичната:

$$K(s) = \frac{1}{1 + e^{-s}} \tag{2.9}$$

Активационна функция се избира според конкретната задача. За класификационни задачи е удачно да се използва стъпаловидна функция, докато за задачи, при които се предсказват непрекъснати стойности, е удачно да се използва сигмоидна или друга подобна непрекъсната функция. Конкретният избор се прави емпирично според постигнатото качество на приближение и цялостната ефективност постигнатата на практика. От теоретична гледна точка няма ограничения за избора при непрекъснатия случай – всяка непрекъсната неконстантна и ограничена активационна функция може да се използва за апроксимация на изхода, стига той да има непрекъсната и частично гладка зависимост от входовете [55, 57, 167].

Според обучаващото правило. Съществуват различни алгоритми за обучение на невронната мрежа. Често е този на обратното разпространение на грешката [152, 153]. Той е подходящ главно за прави невронни мрежи. При него на базата на теглата се изчислява кои неврони най-силно са повлияли на получения резултат, след което на базата на това доколко достоверен е бил резултатът, теглата се намаляват пропорционално на числовата оценка за грешката при класификация описана в 2.1.2.5. Той се опитва да открие с помощта на кои тегла да минимизира тази грешка.

Алтернативи са правилото за обучение на перцептрона [125] и състезателното обучение [127], което наподобява процеса на обучение на естествените неврони предложен от Доналд Хеб в [53]. Друго обучаващо правило взаимствано от правилата въведени от Хеб е споменатата по-рано модификация на алгоритъма за обратно разпространение на грешката, почиваща на биологичните процеси, описана в [17].

2.1.2.5 Обучение с обратно разпространение на грешката

При обучение на невронната мрежа, на входа ѝ итеративно се подава всеки пример от обучаващото множество данни. На всяка итерация алгоритъмът за обратно разпространение на грешката променя теглата на връзките между невроните в мрежата според големината на получената грешка при приложение на модела върху съответния пример. Промяната във всяко тегло w_{ij}^{p}, връзка между неврона p_j с неврон от предходния

слой q_i, се пресмята с помощта на частичната производна на грешката за p_j.

$$\Delta w_{ij}^p = -\alpha \frac{\partial E_j}{\partial w_{ij}^p} = -\alpha \delta_j^p y_i \tag{2.10}$$

Тук α е коефициент, който определя скоростта на обучение на алгоритъма. Той трябва да е малко положително число, например $1/3$. При по-малки стойности на α мрежата ще се приближава по-плавно към търсените оптимални тегла w_{ij}^p, докато при по-големи това ще става по-бързо. С δ_j^z е означена тази част от производната, която не зависи от i.

Грешката е средното квадратично разстояние между получения изход от мрежата и очаквания изход за обучаващото множество. Ако z_j са изходите на невронната мрежа, а z_j^* са очакваните изходи, то грешката има следния вид.

$$E_j^z = \frac{1}{2}\left(z_j - z_j^*\right)^2 \qquad\qquad E_j^{z\prime} = z_j - z_j^* \tag{2.11}$$

При използване на логистична активационна функция, промяната на теглата w_{ij}^z между изходните неврони z_i и невроните от скрития слой y_i може да бъде пресметната по следния начин:

$$K'(s) = \left(\frac{1}{1+e^{-s}}\right)' = K(s)(1-K(s)) \tag{2.12}$$

$$\begin{aligned} \frac{\partial E_j^z}{\partial w_{ij}^z} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial s}\frac{\partial s}{\partial w_{ij}^z} &= \left(z_j - z_j^*\right)\left(K(s)(1-K(s))\right) w_{ij}^y \\ &= \left(z_j - z_j^*\right)\left(z_j(1-z_j)\right) y_i = \delta_j^z y_i \end{aligned} \tag{2.13}$$

Промяната в теглата за изходния слой придобива следния вид:

$$\Delta w_{ij}^p = -\alpha \delta_j^z y_i = -\alpha z_j (1-z_j)(z_j - z_j^*) y_i \tag{2.14}$$

Тъй като за невроните от скрития слой няма очаквани стойности и не може да се пресметне действителна грешка, разликата между очакван и получен сигнал е заменена с линейна комбинация на грешките от следващия слой. Оттам идва името на алгоритъма. Това правило е противоположно на правото разпространението на сигнала

от (2.7).

$$\delta_i^y = \sum_j \left(w_{ij}^z \delta_j^z \right) \tag{2.15}$$

Така промяната в теглата за скрития слой, изчислена с помощта на обратно разпространение на грешката, има следния вид:

$$\Delta w_{ki}^y = -\alpha \delta_i^y x_k = -\alpha y_i (1 - y_i) \sum_j \left(w_{ij}^z \delta_j^z \right) x_k \tag{2.16}$$

При използване на логистична функция целият алгоритъм за обратно разпространение на грешката, приложен за мрежа с един скрит слой изглежда, по следния начин [153]:

Алгоритъм 1. Започваме с невронна мрежа със случайно избрани тегла w_{ij}^p. За всеки пример (x, z^*) от обучаващото множество T:

- Прилага се невронната мрежа върху входните данни x, като се намират стойностите на невроните от скрития слой y, и предсказанията на изходите z.
- Пресмята се Δw_{ij}^z от грешката на изхода на невронната мрежа.
- Пресмята се Δw_{ki}^y с обратно разпространение на грешката.
- Всички тегла в мрежата се променят със съответните стойности на Δw.

Работата на алгоритъма приключва след определен брой обучаващи епохи върху всички примери от обучаващите множество. Този брой може да е фиксиран или да зависи от някакво друго условие – например, обучението може да бъде преустановено, когато се отчете влошаване на качеството на класификация върху отделно валидиращо множество.

2.1.2.6 Приложение на изкуствените невронните мрежи

Невронните мрежи могат да се прилагат в множество от задачи за предсказване на зависимости. Всяка непрекъсната зависимост може да бъде приближена с произволна точност от невронна мрежа с един скрит слой [56]. Това свойство е било доказано

първо при използване на сигмоидна активационна функция [30]. В последствие е било показано, че то е следствие от самия модел на многослойните невронни мрежи, които са изключително изразителен инструмент, независимо от избора на активационна функция – за апроксимация на непрекъсната функция с частично непрекъсната производна може да бъде използвана всяка непрекъсната неконстантна и ограничена активационна функция [57, 55, 167]

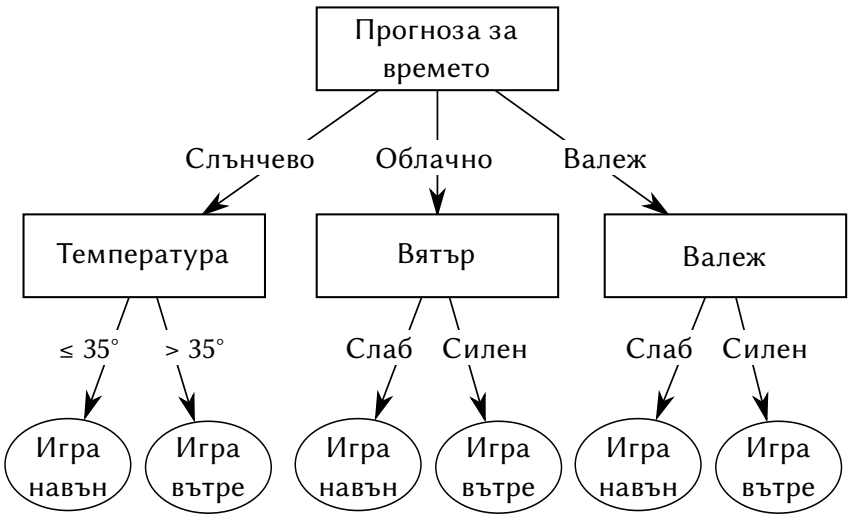
ИНМ представляват мощен инструмент за разпознаване на образи. Прилагат се при разпознаване на текст, глас, жестове и изображения [39], прилагани са успешно при създаване на автономни автомобили [65] и имат огромен потенциал за използване в биоинформатиката, където проблемите често предполагат търсене на сложни взаимовръзки, които са хаотични по характер и не винаги се подават на формализиране и систематизация. Това е особено валидно за задачи, за които няма открити решения извън областта на изкуствения интелект, които да дават достатъчно добър резултат. Така например ИНМ намират приложението си при изследване на болести, които зависят от много фактори, и могат да се използват за откриване на множествата от гени, които влияят на симптоматиката на заболяването [98].

2.1.3 Дърво на решенията и случайна гора

Дърветата на решенията (decision trees) [116] са друг мощен инструмент за машинно самообучение. Те се строят за предсказване на стойността на даден целеви атрибут по стойностите на множество от числови или номинални атрибути. При обхождане на такъв вид дърво, тръгвайки от корена, всеки възел разделя множеството от възможни решения по стойностите на един от атрибутите, докато се стигне до листо, което носи предсказаната стойност на целевия атрибут [123].

Съществуват дървета за извършване на класификационен анализ, които предсказват номинален целеви атрибут, както и дървета за извършване на регресионен анализ, които се опитват да предскажат числовата стойност на целевия атрибут. За разлика от невронни мрежи, които моделират сложни линейни взаимовръзки с участие на всички входни стойности, дърветата на решенията се строят на основата на прости правила върху отделни атрибути, които разделят решенията и създават отделните разклоне-

ния. Това спомага на това те да бъдат изследвани и разбрани от човек. На фигура 2.2 е илюстрирано едно опростено дърво на решенията.



Фигура 2.2: Дърво на решенията

В биоинформатиката, дърветата на решенията са предпочитани пред други модели на машинното самообучение, понеже изследователите могат да изследват причините за взимане на отделните решения [139].

2.1.3.1 Алгоритъм C4.5 за обучение на дърво на решенията

Алгоритъмът C4.5 конструира дърво на решенията според обучаващото множество S, като търси атрибутите, които най-добре класифицират обучаващите примерите. Изборът на атрибут A се основава на информационната печалба $IG(S, A)$ при разделяне на примерите по съответния атрибут.

$$IG(S, A) = E(S) - \sum x \in D(A) \frac{|S_{Ax}|}{|S|} \tag{2.17}$$

Тук $D(A)$ е множеството от стойности на атрибута A, S_{Ax} е множеството от примери, за които атрибута A има стойност x – $S_x = s \in S | A(s) = x$, а $E(R)$ е ентропията на множеството от примери R по целевия атрибут Z.

$$E(S) = - \sum x \in D(Z) P(Z = x) \log_2 P(Z = x) \tag{2.18}$$

На всяка стъпка се избира атрибутът A с най-голямо $IG(S, A)$, който най-добре разделя обучаващото множество S. За разлика от предшественика му ID3, който използва същия подход, C4.5 може да борави с липсващи стойности, както и да работи с непрекъснати атрибути, като разделя множеството, в което се разпростират стойностите им на подинтервали, за които може да се изчисли информационната печалба.

2.1.3.2 Случайни гори

Случайните гори[2] (random forests), наричани още случайни гори на решенията [20], се основават на *ансамблов метод за самообучение* (ensemble method), при който едновременно се конструират множество от дървета на решенията. Всяко отделно дърво прави отделно предсказание, след което статистическата мода на всички отделни предсказания се взема като крайното предсказание на гората. В една случайна гора всяко дърво се построява по случайна извадка от обучаващото множество, като изборът е независим и използва еднакво разпределение за всички дървета.

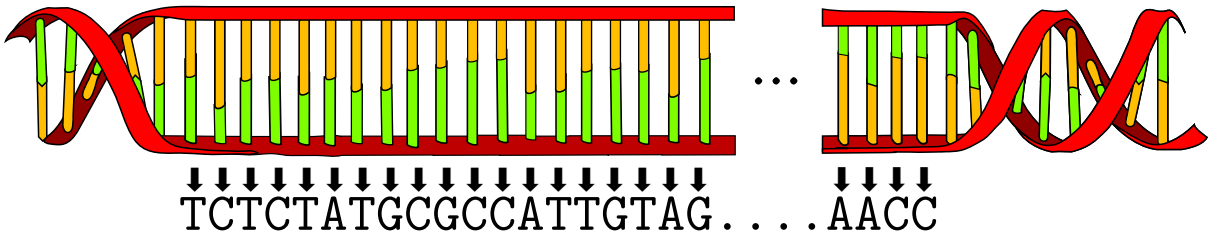
2.2 Анализ на данни от паралелно секвениране

2.2.1 Геномика

При живите организми, ДНК и РНК молекулите са основното средство, което е отговорно за наследяването на кодираните от тях морфологични признаци, определящи облика на биологичните видове. Те са носителите на генетичната информация, която се предава между поколенията и определя цялостното развитие на биологичния индивид. ДНК и РНК молекулите представляват дълги вериги образувани от четири различни нуклеотида, в състава на които участва характеризираща ги нуклеотидна база – аденин, гуанин, цитозин или тимин за ДНК; аденин, гуанин, цитозин или урацил за РНК. Нуклеотидната последователност на тези бази в ДНК веригата кодира генетичната информация и е обособена в отделни значещи единици, наречени гени.

[2] Преводът „случайна гора“ е бил използван от авторите на Intelligent Framework for Risk Management за GMES Operational Capacity Workshop в София, `http://gmes-bg.org/presentations/42_Mr._Roumen_Venkov_Intelligent_Framework_for_Risk_Management.pdf`

По своето естество, и ДНК и РНК са цифров носител на данни, и това силно спомага изследването им със средствата на информатиката. От информатична гледна точка, те представляват последователности над четирибуквена азбука (A, G, C, T)[3], наричани секвенции, в които всяка буква отговаря на един четирите нуклеотиди. На фигура 2.3[4] е показан видът, в който една ДНК последователност постъпва за обработка в компютъра.



Фигура 2.3: Прочит на ДНК последователност в цифров вид

В постъпващите данни, гените представляват подпоследователности, чиито позиции могат да предсказани чрез различни изчислителни методи, но взаимовръзките между тях и смисълът им остават неизвестни без извършване на обстойни научни изследвания, съсредоточени върху отделни гени или групи от гени. Начина, по който те кодират генетичната информация, е естествено неструктуриран и хаотичен, което значително утежнява тези изследвания. Може да се направи аналогия с машинния код, генериран от един модерен компилатор, който влага голяма степен на оптимизация в програмите, но сложността в генетичните вериги е от много по-различно естество.

Паралелното секвениране е съвременен технологичен процес, при който от всяка ДНК или РНК проба могат да се прочетат в електронен вид значителен брой къси нуклеотидни последователности, които впоследствие с помощта на изчислителните средства на биоинформатиката биха могли да бъдат навързани така, че да бъде възстановена тяхната оригинална последователност. За нов вид, който не е бил секвениран, това е значително скъп и трудоемък процес. При някои видове този процес може да бъде допълнително усложнен от биологични фактори, както например е при пшени-

[3] Биологически, в РНК базата T е заменена с еквивалентната база U, но това няма отношение към тук разглежданите изчислителни задачи

[4] Тази фигура използва графични елементи взети от илюстрация на ДНК репликация на Mariana Ruiz – `http://commons.wikimedia.org/w/index.php?title=File:DNA_replication_en.svg&oldid=109410354`

цата, чийто геном е мултиплициран в три сходни, но различни свои копия.

След прочитането на генетичните данни, те могат да бъдат използвани за различни изчислителни изследвания. Откритите вече гени могат да се използват за намиране на зависимости при търсене на нови лечения на различни заболявания, за създаване на нови лекарства на основа на тяхната структура, в генното инженерство и други. Дори отделни отрязъци от ДНК намират широко приложение, например като маркери за идентификация в съдебната медицина [64], за откриване на еволюционните връзки между видовете и други.

2.2.2 Метагеномика

Метагеномиката (metagenomics) е област от геномиката, която се занимава с изследване на генетичен материал от микроорганизми, взет от проби, събрани в хетерогенни биологични среди. Такива среди могат да включват почва, водни басейни, както и вътрешностите на различни по-висши в еволюционно отношение организми. Съобществата от микроорганизми, които ги обитават, са слабо изучени, като дори оценката за разнообразието на видовете подлежи на задълбочено и всеобхватно изследване.

Основните задачи на метагеномиката са сравнителен анализ на микробиалните общества за оценката на биоразнообразието и проучване на филогенетичните (phylogenetic) взаимовръзки, които имат значение за различни научни сфери, свързани с човешкото здраве [105], земеделието [77], бактериалната и вирусната еволюция [73] и еволюцията на видовете като цяло.

Като най-бързо мутиращи видове в природата, бактериите и вирусите са подходящи за изследване на еволюционните процеси отблизо, а в същото време са и основна причина са избухването на епидемии, както сред хората, така и сред селскостопанските животни и земеделските култури. Поради това изследването на еволюцията им е от съществена важност за човешкото знание, както при изучаването на миналото на биосферата – започнала от биологични форми много сходни с живеещите днес микроби, така и за бъдещето на човечеството – в частност, борбата с епидемии, справянето с антибиотичната резистентност и други.

За разлика от геномните изследвания на отделни организми, тук данните имат

много по-различен вид. Обикновено се изследват близки или съвпадащи отрязъци от ДНК последователностите на много различни видове. За тези последователности изключително рядко са налични референтни бази от данни, които да улеснят процеса на ресеквениране. В същото време не е възможно да бъдат култивирани отделни организми, за да бъдат обособени за изследване. Поради това се работи с множества от данни с хетерогенен състав, включващи много различни видове, и съдържащи много висока естествена вариация. По тази причина, както изследванията, така и трудностите които един изследовател може да срещне, се различават значително от тези в геномиката, и се налага използването на различен набор от подходи и инструменти.

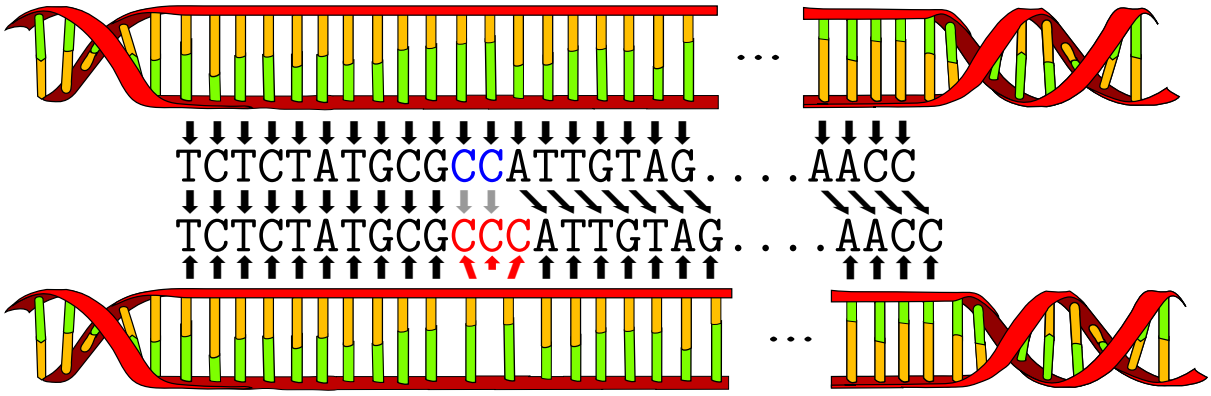
Поради това, че в голямата си част тези инструменти не са претърпели достатъчно дълго развитие, изследванията в метагеномиката могат срещнат допълнителни трудности [144]. След използването на процедурата по паралелно секвениране, всяко метагеномно изследване е свързано с обработката на голям обем от данни без известни взаимозависимости. Рядко може да се разчита на допълнителни помощни знания от генетични бази от данни, в които присъстват само малка част от изследваните видове. Налага се работа с наличните към момента софтуерни инструменти, които не са достатъчно развити, и се основават на алгоритми, които изискват значителни изчислителни ресурси. За отделните фази на изследването се налага избора измежду различни инструменти, чиято пригодност към извършвания експеримент е не винаги предварително известна [133, 34].

2.3 Приложение на информатиката в геномните изследвания

2.3.1 Алгоритми за съпоставяне

При обработката и изследването на генетични данни, получени в процеса на секвениране, често се налага да се прибегне до процедура по *съпоставяне* (alignment). Това е необходимо при изпълнение на алгоритми за сравнение на *секвенциите* (последователностите, sequences) от близки видове, чието действие бива нарушено, ако функционалните части са отместени поради вмъкнати или изтрити бази в нуклеотидните

вериги, наричани инсерции и делеции, които са част от естествената междувидова вариация. Това е проблем както при изследването на ДНК и РНК секвенции, така и при изследването на протеинови секвенции.



Фигура 2.4: Погрешно вмъкване при процеса на секвениране

Отмествания от същия вид могат да се появяват и като грешки при самия процес на секвениране, често при погрешна оценка на броя *мононуклеотидни повтори* (последователно повтарящи се бази). Това може да затрудни откриването на идентични фрагменти и последващо асемблиране или анализ, поради което се налага суровите данни също да бъдат подложени на процедура по съпоставяне. Илюстрация на вмъкната база в следствие на грешен прочит при наличие на повтаряща се база е показана на фигура 2.4.[5]

Съпоставяне се налага и при обработката на сурови данни, където отместванията предизвикани от грешки на секвениращия апарат могат да затруднят откриването на идентични фрагменти, както и техния последващ анализ или асемблиране. Този процес се използва както при обработката на ДНК и РНК секвенции, така и при изследването на протеинови последователности.

Съпоставянето е оптимизационна задача, при която в последователностите постепенно се вмъкват *интервали* (gaps), за да се минимизира някаква оценка за разстоянието между тях. Целта е последователностите да бъдат сближени възможно най-много с възможно най-малко вмъкнати интервали. Това се постига чрез минимизация на

[5] Тази фигура използва графични елементи взети от илюстрация на ДНК репликация на Mariana Ruiz – `http://commons.wikimedia.org/w/index.php?title=File:DNA_replication_en.svg&oldid=109410354`

целева функция, която наказва както отделните разлики, така и на всеки вмъкнат интервал. Особено при съпоставянето на протеинови последователности, цената на всяка разлика се взема от т.нар. матрица на субституциите [2], която съдържа оценка за различието между две аминокиселини (или две бази), които са изчислени емпирично [33, 54].

2.3.1.1 Видове съпоставяне

Съпоставянето може да бъде локално или глобално. Алгоритмите за глобално съпоставяне се опитват да наложат двете последователности по цялото им протежение, докато алгоритмите за локално съпоставяне се опитват да открият сходни подпоследователности върху които те да се съсредоточат. Локалното съпоставяне е много по-точно, когато се борави с фрагменти, които покриват различни части от цялата секвенция и се припокриват само частично, но обикновено е по-сложно за изчисление.

Също е важно и делението на алгоритми, които се прилагат върху двойки секвенции (pairwise alignment), и алгоритми за *множествено съпоставяне* (multiple alignment), които се прилагат върху три или повече секвенции. Намирането на оптималното съпоставяне по двойки е изпълнима задача, и се прилага когато е налична референтна секвенция, към която да бъдат съпоставени останалите. Когато се извършва асемблиране на нов геном или се извършва анализ без наличие на референтен геном е необходимо да бъдат съпоставени по-големи групи.

За съжаление намирането на оптималното съпоставяне на произволен брой секвенции е NP-пълна задача [148]. При асемблиране на нови геноми бройката на секвенции покриващи един участък от ДНК е достатъчно ограничена, за да позволява използването на алгоритми с високо качество. За сметка на това в метагеномиката се изследват значителен брой различни организми с високо ниво на вариация, за които не съществуват референции, което е пряко отразено и в големите множествата от данни. Това налага прилагането на неточни алгоритми.

2.3.1.2 Алгоритъм на Нийдълман и Вюнш

За намирането на оптималното глобално съпоставяне на двойка секвенции се използва алгоритъмът на Нийдълман и Вюнш [104]. Освен в биоинформатиката, той се из-

ползва и в социологията за откриване на оптималното съответствие между хронологически подредени последователности от наблюдения върху социоикономическото състояние на двойки индивиди [1].

При два символни низа с дължини n и m се обхождат възможните съответствия между двойките позиции в двата низа (i, j), и се търси оптималния път от началото на двата до края на двата низа. Използва се рекурсивна дефиниция на матрицата F, чийто елементи F_{ij} съдържат оптималната стойност за съпоставяне на първите i и първите j букви от двата низа съответно.

$$F_{0j} = dj, F_{i0} = di \tag{2.19}$$

$$F_{ij} = \max(F_{i-1,j-1} + S(A_i, B_j), F_{i,j-1} + d, F_{i-1,j} + d) \tag{2.20}$$

Тук d е цената за вмъкване на интервал, първия аргумент на max съответства на запазване на съответствие, втория на добавяне на интервал в първия низ, а третия на добавяне на интервал във втория низ. Използва се динамично оптимиране с кубична сложност за изчисление на F_{nm} като интервалите се вмъкват на съответните места според избрания път при изчисление на всяко F_{ij}.

Алгоритъма е еквивалентен на изчисление на разстоянието на Левенщайн [170], което е формулирано като минималния брой редакции необходими, за да се превърне един символен низ в друг и използва аналогична рекурсивна дефиниция. То се изчислява с подобния алгоритъм на Вагнер и Фишър [147], като съществуват още няколко независимо разработени алгоритъма за извършване на същата процедура [102].

2.3.1.3 Прогресивно множествено съпоставяне

Като NP-пълна задача, намиране на оптималното множествено съпоставяне за по-големи множества от данни е невъзможно, което налага използването на неточни методи. Една група алгоритми използват така нареченото *прогресивно* съпоставяне (progressive alignment). При тях се използва водещо дърво (guide tree) построено с йерархично клъстериране (разгледано в 2.3.3.2) и двойките секвенции се съпоставят една по една до достигане на корена на дървото [99].

Тук е схематизиран стандартният алгоритъм за прогресивно съпоставяне, който има изчислителна сложност от $O(n^3)$.

Алгоритъм 2. Множествено съпоставяне с помощта на водещо дърво

1. Построява се матрица на разстоянията между двойките секвенции, използвайки разстояние на Хеминг (по подобие на 2.3.3.3), пресметнато след съпоставяне по двойки, например с алгоритъма на Нийдълман и Вюнш. Поради разгледаните в 2.3.1.2 особености това дава същия резултат като директно приложение на разстоянието на Левенщайн.

2. С помощта на йерархично клъстериране, като сливането на съседи от 2.3.3.5, се построява двоично водещо дърво. Това дърво е сходно с филогенетичните дървета от 2.3.3.2 и използва идентични алгоритми за построяване, но понеже е изградено на основната на съпоставяне по двойки вместо множествено съпоставяне, не може да се използва като такова. Това е стъпката от алгоритъма със сложност $O(n^3)$, тъй като това е сложността на алгоритъма за сливане на съседи.

3. Тръгвайки от листата на дървото, всяка двойка секвенции се съпоставят една с друга, и се заместват с една обобщена секвенция-образец (pattern), която представя статистически съдържанието и на двете, след което двойките секвенции-образци се съпоставят като обикновени секвенции, докато се стигне до корена.

4. Интервали (gaps), добавени към секвенциите и образците по време на съпоставянето на двойките, се пренасят надолу по дървото и се добавят към оригинална секвенция за получаване на крайното съпоставяне получено след събиране на всички добавени интервали по всеки клон на дървото.

Една софтуерна имплементация на този подход е Clustal [28] и неговата популярна версия ClustalW [142]. Като относително пряка реализация на алгоритъм 2, той е популярен инструмент за обучение и обект на много модификации, включително и за паралелизация [27, 94, 118, 19].

Основният недостатък на прогресивното съпоставяне е, че грешката при обхождане към корена на дървото се натрупва [99]. Една от причините за това е съпоставането

по двойки, което се извършва както при изчисляването на матрицата на разстоянията в стъпка 1, така и при извършване на прогресивната стъпка 3. То не винаги съответства на коректното съпоставянето на целия масив от последователности. Съществен проблем представляват и грешките в прочитите, чийто ефект също се натрупва при обхождане на дървото.

Съществуват различни решения на този проблем. Например T-Coffee [106] се опитва да намали грешката, като включва и други секвенции при извършване на прогресивната стъпка, но това сериозно намалява производителността. Друг вариант е използването на полу-прогресивния алгоритъм на PSAlign [141], който се основава на различна формулировка на задачата за съпоставяне, която не е NP-пълна и може да бъде решена точно и без евристиките, които внасят шум.

2.3.1.4 Итеративно множествено съпоставяне

Основният недостатък на чисто прогресивните подходи е липсата на последваща обработка за подобряване след приключване на прогресивния етап от алгоритъма, резултатите от който силно зависят от качеството на първоначалното съпоставяне. Този проблем е решен в *итеративните* подходи, които изпълняват прогресивна процедура по съпоставяне, след което итеративно се опитват да подобрят качеството [99].

Това води до откриване на съпоставяне, значително по-близко до оптималното, а в голяма част от случаите води и до значително повишаване на производителността понеже качеството на използваните евристики при прогресивния етап е от по-малко значение за крайния резултат, което позволява те да бъдат заменени с още по-опростени процедури. Това позволява обработката на много по-големи масиви данни, съдържащи хиляди последователности.

Популярен инструмент за итеративно съпоставяне е MUSCLE [38], който осъществява два етапа на прогресивно съпоставяне, вторият от които използва еволюционното разстояние на Кимура [68], след което извършва няколко етапа на подобряване (по подразбиране, най-много 16). Негова алтернатива е MAFFT [67], който използва бързо преобразуване на Фурие [84] както за втория прогресивен етап, така и за итеративното подобряване.

2.3.2 Асемблиране

Асемблирането (genome assembly) е процес, при който кратките прочити, получени след паралелно секвениране, биват съпоставени и обединени в дълги последователности, наречени *контиги* (contigs), като в голяма част от случаите крайната цел е да бъде асемблиран целият геном. Не съществуват технологии, които да прочетат целите ДНК вериги, нито дори части със значителна дължина. Паралелното секвениране, което е в класа на най-производителните технологии, генерира кратки фрагменти с дължини, които не надминават 1000, дори в повечето случаи 100-150. Това налага използването на алгоритми за асемблиране [143, 92, 111, 101].

Асемблирането се използва при *de novo* секвенирането на неизвестни геноми [78], както и при ресеквениране на вече известни геноми. В болшинството от случаите то е необходима първа стъпка преди извършаването на какъвто и да било анализ, включително геномно анотиране [138], откриване на точкови мутации (SNPs) [110]. Използва се и при анализ на геномите на известни бактерии [15, 100], въпреки че те в голямата си част разчитат на метагеномни изследвания, които се налага да бъдат извършвани без налични асемблирани геноми [61, 144].

Един клас алгоритми за асемблиране са тези на припокриване–разположение–консенсус. При тях се откриват припокриванията между прочитите, след това те се поставят в граф, който описва тяхното местоположение и по него се изгражда консенсусната секвенция. Алтернатива е използването на графи на де Брюн, при които кратките прочити се разбиват на k-мери (поднизове с дължина k) преди поставянето им в графа. Те са по-подходящи за кратки прочити с високо покритие, но са неподходящи, ако използваната технологията генерира дълги прочити, които имат по-слабо покритие [81, 95, 132, 44].

Реализация на първия подход е използвана в ARACHNE [12], който е бил популярно решение за по-старите технологии за секвениране разчитащи на последователно четене, при които прочитите са по-дълги. Графите на де Брюн се използват във Velvet [164]. В [112] е представен алтернативен подход за асемблиране, основан на търсене на ойлерови пътища. SSAKE [149] и SHARCGS [35] разчитат на таблица за търсене, индексирана по префикса на прочитите.

Съществуват и решения като Ray [18], които асемблират пълната последователност с различни алгоритми и после се опитват да открият консенсус между тях. Изследването GAGE [131] има за цел да предложи стандартен подход за оценяване и сравняване на различните подходи за асемблиране.

Асемблиране и съпоставяне

Голяма част от алгоритмите за асемблиране извършват някаква форма на съпоставяне, обикновено след извършване на клъстериране. Тези алгоритми за съпоставяне се различават значително от използваните за множественото съпоставяне, обсъдени по-рано, поради еднородния характер на клъстърите. Bowtie [76] е пример за софтуер, който може да се използва за решаване на тази задача. Той е едно от решенията, които разчитат на популярната Burrows-Wheeler трансформация [44].

2.3.2.1 Графи на де Брюн

Графите на де Брюн описват всички припокривания с дължина n на последователности над m-буквена азбука. Всяко припокриване е представено с връх в графа, който има m^n върха. Тези графи са били независимо въведени неколкократно за различни теоретични цели в математиката. Де Брюн [24] ги използва, за да докаже хипотеза за броя на циклите P_n от 2^n двоични цифри с неповтарящи се подпоследователности с дължина n, които са също и най-кратките цикли, съдържащи всички възможни подпоследователности с дължина n. Тази задача възниква по-рано от проблем с телекомуникациите. Гуд [49] ги прилага за конструирането на повтарящи се десетични числа с нормалност от ред r, тоест всички последователности от цифри с дължина r с нормална честота на цифрите от 10^{-r}. Много по-рано техните свойства са били използвани без формална дефиниция в [128].

Графите на де Брюн са обобщени за азбуки с големина повече от 2, както и за произволни речници, които не съдържат непременно всички последователности на използваната азбука. Такъв е случаят в геномиката, където те се използват за описание единствено на припокриванията между последователностите, които съществуват в изследваното множество данни.

Откриването на най-доброто асемблиране на фрагментите може да се извърши с

решение на задачата за търсене на Хамилтонов цикъл. На всеки връх на графа се съпоставя k-мер от изследваните последователности и се търси Хамилтонов цикъл, който минава през всички k-мери. Такова решение е било успешно приложено за откриване на първия бактериален геном [43]. Така обаче задачата става NP-пълна и това сериозно затруднява приложението на модела поради изключително високите изисквания към изчислителните ресурси.

Графите на де Брюн позволяват асемблирането да се търси с решаване на много по-проста задача. Там k-мерите съответстват на ребра, а не на върхове. Конструира се граф на де Брюн, във който всеки връх е суфикс или префикс на някой k-мер и има дължина $k-1$, като върхът за всяка последователност може да се среща само веднъж. За всеки k-мер се построява ребро между неговия префикс и суфикс, като на реброто съответства самият k-мер. По този начин задачата се редуцира до търсене на Ойлеров цикъл [29].

2.3.3 Йерархично клъстериране и филогенетични дървета

2.3.3.1 Филогенетични дървета

Филогенетичните (или еволюционните) дървета представляват таксономия на биологичните видове, която има за цел да опише вероятните еволюционни връзки между тях. Листата на едно филогенетично дърво са изследваните биологични видове. В някои случаи, особено в метагеномиката, където са налични ДНК последователности без допълнителна информация за самите видове, листата могат да бъдат оперативни таксономични единици, които отговарят на все още неизследвани видове или групи от видове [16]. Всички вътрешни върхове на едно филогенетично дърво са хипотетични таксономични единици, които отговарят на предсказан неизвестен прародител или група прародители на върховете наследници.

Практиката за класификация на видовете съществува още от най-ранните документирани периоди на човешката история. Още в Древен Египет е била практикувана класификация на лечебните растения. Аристотел първи се опитва да извърши пълна класификация на всички живи същества, която включва много категории, които все още присъстват в модерните класификации – като например разделението на гръб-

начни и безгръбначни [88]. Неявна класификация съществува и като част от говоримия език.

Създаването на модерни таксономии започва през 16-ти век с трудовете на Андреа Чезалпино, в които той извършва класификация на 1500 ботанически вида на базата на множество фактори, включително формата на техните семена и плодове, и начина им на израстване. Работата на много други ботаници в следващите векове завършва с изграждането на таксономната система на Карл Линей, която продължава да се използва и в наши дни в почти непроменен вид [82, 83, 88].

Всички изброени опити за класификация са основани на външните белези на на видовете, което е голяма част от причината много от тези ранни класификации в много случаи да са значително отдалечени от действителните еволюционни разстояния между видовете, които присъстват в тях. Чарлз Дарвин първи споделя идеята, че зад таксономните групи стоят родословни дървета, които съдържат неизвестни древни прародители [32]. Заедно с това той споделя и виждането си, че всички такива дървета имат общ корен. Въпреки това откриването на действителните връзки между видовете не започва до откриването на молекулата на ДНК [93], изучаване на структурата ѝ [71, 72, 66, 70] и откриване на ролята ѝ в наследяването на биологични признаци [169, 168, 5].

ДНК секвенирането позволява използването на много по-добри математическо обосновани оценки, които дават много по-точна представа за еволюционното разстояние между всеки два вида. Това води до множество открития за неочаквани групирания и разделения в таксономията, която на места претърпява значително преструктуриране. Чрез ДНК анализ е установено, че шимпанзето е животинския вид най-близък до човека [90], макар и морфологичните сравнения да подкрепят или идеята, че всички човекоподобни маймуни са еднакво отдалечени, или че човека е по-близък до друг вид човекоподобни маймуни. По същия начин, през 70-те години е установено, че археите формират отделно надцарство от животински видове, различно от бактериите [157].

ДНК секвенирането в крайна сметка спомага и за развитието на най-модерния подход към биологичната класификация – кладистиката, от гръцки „кладос“ (κλάδος, клон), наричан още филогенетична систематика. При нея видовете не използват кла-

сическите таксономни разделения, а кладовете са изградени директно на основата на предсказаното генетично сходство, и оттам очакваните им прародители. Така например вместо просто като четириноги хордови животни, птиците (aves) биват класифицирани по-точно към клада paraves, обхващащ всички птицеподобни динозаври[6] [115, 159, 48]. При кладистиката класификацията се основава на научно проверяема хипотеза – че видовете подставени в един клад са по-близки помежду си отколкото до които и да било други видове извън клада [22]. Близостта се измерва на базата на еволюционно разстояние, което е броя поколения, които разделят два вида, но в голяма част от случаите може да се замени с евристично изчислени приближения сметнати на базата на генетичните различия.

Построяването на дървета на еволюцията на базата на ДНК анализ има особено значение за метагеномиката, където се изследват микроорганизми и в голяма част от случаите практически не съществуват алтернативни методи. Значителна част от голямото разнообразие на микроорганизми остава скрито [103]. Освен че изолирането на отделни видове и изследване на техните външни белези е неприложим подход, дори със средствата на ДНК анализа изследването на разнообразието им е тежка задача. Съществуват значителни пречки те да бъдат култивирани в лаборатория, което налага те да бъдат изследвани в проби от естествената среда, поради което значителна част от дървото на организмите остава неизследвано [121]. Високата скорост на мутация [37] допълнително затруднява точното отличаване на видове.

2.3.3.2 Йерархично клъстериране

Съществуват различни подходи за построяване на филогенетично дърво с помощта на данните от определена ДНК проба. Една голяма група от тях са основани на йерархично клъстериране на базата на разстояние. Това включва сливането на най-близки съседи (neighbour joining) [129] и подобния на него UPGMA [137].

[6] Отчитайки последните изследвания [48], подробна класификация на птиците би изглеждала по следния начин: Avetheropoda (птицетероподи) – Coelurosauria (целурозаври; „гущероподобни с куха опашка“, б. а.) – Tyrannoraptora (тиранораптори) – Maniraptoriformes (манираптообразни; включва всички пернати динозаври, б. а.) – Maniraptora (манираптори) – Pennaraptora (пенараптори; „пернати раптори“, б. а.) – Paraves (параптици; птицеподобни, б.а.) – Eumaniraptora (еуманираптори) – Avialae (птици)

И при двата подхода се построява матрица на разстоянията между двойките секвенции, и после най-близките се сливат итеративно в процедура по натрупващо се клъстериране, като на всяка стъпка се присъединява нов възел към дървото, свързва се с ребра с предсказани дължини, и това продължава докато се свържат всички възли. При UPGMA се открива корена на дървото, но се разчита на предположението, че скоростта на еволюцията е равномерна. При сливането на съседи вместо това се прави предположението, че кумулативната еволюция е минимална, но построеното дърво е с неизвестен корен.

За дърветата с неизвестен корен съществуват различни подходи, които позволяват корена да бъде избран впоследствие, но в общия случай това не е възможно без да е налична информация за скоростите на разклоняване на клоновете на дървото. Молекулярният часовник, определящ скоростта на еволюция, както и генетичното разстояние между видовете, биват обект на различни изследвания. Тръгвайки от хипотезата, че между тях има зависимост [165, 89], могат да се приложат алгоритми, използващи разстоянията между самите последователности [99]. По-нови изследвания обаче поставят тази хипотеза под въпрос [58], оставяйки възможност да бъде използвана топологията на самото дърво.

Недостатък на всички алгоритми, основани на разстояние, е, че те по-трудно отличават *хипервариабилни участъци* (hypervariable regions), които се срещат в повече от един клон на дървото и внасят голяма разлика между най-близките му членове. Съществуват алтернативни методи, при които този проблем не съществува, но не могат да се похвалят със същата производителност. Примери за това за методите на максималната икономия и използване на метода на максималното правдоподобие [42, 166].

Методът на максималната икономия разчита на същото предположение като сливането на съседи – че еволюцията с най-малко мутации е най-правдоподобна. Вместо да се разчита на разстоянията между секвенциите се работи направо с отделните позиции в тяхното съпоставяне. Дърветата се конструират на базата на минималния брой мутации, които са необходими да се достигне от една секвенция до друга, като се търсят дървото, за които цената на еволюцията е най-малка [154, 11]. Откриването на оптималното дърво е NP-пълна задача, но съществуват различни приближени алгоритми.

Един основен недостатък тук остава предположението, че съвпадение в генетичния код на два вида винаги е следствие от това, че той е бил наследен от общ прародител. Това не е непременно вярно, понеже е възможно той да е бил получен в следствие на хоризонтален пренос на гени [113], сходимост на генетичните последователности [108, 11] и дори случайно.

Алтернатива на максималната икономия е използването на метода на максималното правдоподобие. При него се изчислява правдоподобието на всяко дърво на основата на вероятността отделните мутации в него да се случат, след което се извършва търсене в пространството на дървета, което да намери дървото за което правдоподобието е най-голямо [154, 166]

2.3.3.3 Матрица на разстоянията

Подходите за йерархично клъстериране разчитат на използване на матрица на разстоянията $d(x, y)$, която служи за откриване и сливане на клъстърите. Такива матрици бяха споменати накратко при разглеждане на подходите за прогресивно множествено съпоставяне в 2.3.1.3, тъй като тези подходи също разчитат на изграждане на дървета с помощта на идентични алгоритми за клъстериране.

Основната разлика е, че самото множественото съпоставяне се извършва преди такова съпоставяне да е вече налично, докато за да бъде построено на коректното филогенетично дърво то е абсолютна необходимост. И докато в първия случай матрицата може да се изгради на базата на по-неточен метод измерващ разстоянията между секвенции съпоставени по двойки, при същинските филогенетични дървета трябва да се използва точното разстояние, което е пресметнато наново след като вече е приключила пълната процедура по множествено съпоставяне.

Това означава също, че на този етап вече се използва само разстоянието на Хеминг. Споменатите по-сложни метрики за разстояние, които допринасят значително при решаването на други задачи, тук вече не са необходими и дори биха могли да влошат резултата. Това е така понеже тези метрики са въведени, за да компенсират разминаването между секвенциите, което вече е било елиминирано с извършването на пълната процедура по намиране на оптималното множествено съпоставяне.

Така матрицата на разстоянията се попълва по следната формула.

$$d(x,y) = \frac{1}{m}\sum_{i=1}^{m}[x_i \neq y_i] \tag{2.21}$$

2.3.3.4 UPGMA

UPGMA е съкращение на unweighted pair-group method using arithmetic averages, или непретеглен метод за групиране на двойки използващ средни аритметични. Той използва средното разстояние между двойките клъстъри, за да ги слива итеративно, докато всички секвенции се намират в един клъстър [137, 166].

UPGMA разчита, че във всеки един момент двата най-близки клъстъра са тези, които са и еволюционно най-близки. На всяка стъпка се попълва липсващия между тях възел в дървото, за който метода предполага, че трябва да бъде еднакво отдалечен и от двата. При сливането на последните два клъстъра се построява възелът, който се избира за корен.

За изпълнение на метода първо се въвежда средното разстояние между два клъстъра $\mathcal{X}$ и $\mathcal{Y}$ по следния начин.

$$d(\mathcal{X},\mathcal{Y}) = \frac{1}{|\mathcal{X}||\mathcal{Y}|}\sum_{x\in\mathcal{X},y\in\mathcal{Y}} d(x,y) \tag{2.22}$$

След това се изпълнява следният алгоритъм.

Алгоритъм 3. Итеративно сливане на клъстъри

1. Всяка секвенция x се поставя в едноелементен клъстър $\mathcal{X}$, а в еволюционното дърво се добавя листо, съответстващо на съответния клъстър.
2. Докато броят на клъстърите е по-голям от 1:
   (а) Намира се двойката $\mathcal{X}$, $\mathcal{Y}$, за която разстоянието $d(\mathcal{X},\mathcal{Y})$ е най-малко.
   (б) Клъстърите $\mathcal{X}$ и $\mathcal{Y}$ се сливат в клъстър $\mathcal{Z}$.
   (в) В дървото се прибавя възел съответстващ на новия клъстър, който се свързва с възлите на $\mathcal{X}$ и $\mathcal{Y}$. Дължината на дъгите е $\frac{1}{2}d(\mathcal{X},\mathcal{Y})$.

Трябва да се отбележи, че на всяка стъпка изчисляването на разстоянието между новополучения клъстър $\mathcal{Z}$ и всеки друг клъстър $\mathcal{W}$ може да бъде опростено като се използва направо следната формула.

$$d(\mathcal{Z}, \mathcal{W}) = \frac{|\mathcal{X}|d(\mathcal{X}, \mathcal{W}) + |\mathcal{Y}|d(\mathcal{Y}, \mathcal{W})}{|\mathcal{X}| + |\mathcal{Y}|} \tag{2.23}$$

Този метод е относително лесен за приложение и реализация. Основният му недостатък е начинът, по който се избира реда на клъстърите, определящ и избора на корен. Той разчита на предположението, че еволюцията протича с еднаква скорост – тоест, че натрупаните разлики за единица време ще бъдат едни и същи в различните клонове на дървото. От това предположение следва, че двата клъстъра, които са най-близки, са се отделили най-скоро хронологически, и че за корен правилно е избран последният клъстър, който се е разделил най-отдавна. За съжаление, макар и това предположение често да е приемливо [107], то далеч не винаги е валидно [36, 135]. Това е още по-голям проблем особено когато видовете, условията или средата, която те са обитавали, са различават значително [6].

2.3.3.5 Сливане на съседи

По подобие на UPGMA, сливането на съседи [129] изгражда дървото от листата към корена. При него не се разчита на предположението за равномерна скорост на еволюция, с което се решава проблема със случаите, в което то не е било изпълнено. Вместо това се разчита на предположението за минимална еволюция – търси се дървото, в което общата дължината на клоновете е възможно най-малка, което съответства на случая, в което в различните еволюционни разклонения има възможно най-малкия брой мутации и съответно възможно най-малката сложност [166].

За да се използва общият брой мутации, в алгоритъма се използва общото разстояние $r(x)$ на всеки отделен възел от всички останали възли.

$$r(x) = \frac{1}{n-2} \sum [y \neq x] d(x, y) \tag{2.24}$$

Тук n е общият брой на секвенциите в множеството. С помощта на $r(x)$ се въвежда

и нова матрица Q, в която общото разстояние е извадено.

$$Q(x,y) = (n-2)(d(x,y) - r(x) - r(y)) = (n-2)d(x,y) - \sum_z d(x,z) - \sum_z (z,y) \quad (2.25)$$

След построяване на матрицата Q се прилага следната процедура по сливане на възли, докато всички възли не бъдат слети в общо дърво.

Алгоритъм 4. Сливане на най-близките възли

1. Намират се двойката разлчни x и y, за които $Q(x,y)$ е най-малко.
2. x и y се сливат в нов възел z, който се присъединява към дървото и се свързва с x и y. Дължините на дъгите се пресмятат с (2.26) и (2.27).
3. Възлите x и y се премахват от Q и са заменят със z. С помощта на 2.28 се пресмятат дължините до новия възел и се преизчислява Q.

Дължините на новопостроените ребра δ в дървото се пресмятат по следната формула.

$$\delta(x,z) = \frac{1}{2}d(x,y) + \frac{1}{2(n-2)}\left(\sum_w d(x,w) - \sum_w d(y,w)\right) \quad (2.26)$$

$$\delta(y,z) = d(x,y) - \delta(x,z) \quad (2.27)$$

Новите разстояния d между новия възел z и всички останали възли в дървото се смятат по следната формула.

$$d(z,w) = \frac{1}{2}\left(d(x,w) + d(y,w) - d(x,y)\right) \quad (2.28)$$

Резултатът от изпълнение на така описания алгоритъм е едно дърво с неизвестен корен. Популярен подход за откриване на корена е използването на външна група, която е еднакво отдалечена от всички видове в дървото. Такава възможност съществува, например, в един от първите софтуери за филогенетичен анализ предлагащ сливане на съседите – PHYLIP [41]. Последните изследвания показват, че еднаквата отдалеченост от външната група не зависи от скоростта на еволюция [58].

Освен че се използват приближени методи, разгледаните тук подходи разчитат на голям брой предположения. Някои от тях – като например това за връзката между скоростта на еволюцията и отдалечеността от външна група [58] може да са погрешни. Освен пряко, процесите на еволюция могат да се изследват и като бъдат моделирани с математически средства [10], което представя още една възможност да бъдат изгладени недостатъците в разбирането им.

Проблем остават и големия брой грешки, които се допускат в процеса на секвениране [63]. Шума внесен от оборудването може да доведе до нарушаване на анализа на еволюцията и биоразнообразието [74].

2.3.4 Откриване и корекция на грешки

Преди търсене на оптималното съпоставяне[7] между прочитите и преди и извършването на същински анализ, като построяване на филогетично дърво, се налага да бъдат изчистени зашумените данни. Съществуват различни практики, които включват премахване на прочити с очаквано ниско качество и подрязване на всички прочити до определена дължина, които се прилагат винаги преди извършването на анализ, но дори след тях остават голям брой неоткрити грешки в данните. Този труд е съсредоточен върху разработката на методи за откриване на тези грешки.

Тук са разгледани различни средства, които се занимават с проблема на грешките в кратките прочити, получени при паралелно секвениране.

SHREC. Алгоритъмът за корекция на грешките SHREC [134] използва *обобщено суфиксно дърво* (generalised suffix trie) за откриване на грешките при прочит на нови геноми. Открива над 96% от погрешните прочити и постига точност на корекцията от над 80%. Той е насочен към корекцията на грешки получени в сравнително хомогенните данни, които се срещат при секвениране на геноми на един организъм, и не взема предвид спецификите, които се срещат в едно метагеномно множество от данни. Освен това е изграден и тестван с данните прочетени със апарат за секвениране на

[7] За разлика от други подходи, методите за корекция на грешки предложени в този труд разчитат на вече налично съпоставяне, което означава, че корекцията на грешки се извършва след, а не преди, съпоставянето. При нужда, то може да се повтори с коригираните данни.

Illumina, който генерира значително по-кратки прочити [86] от изследваните в тази работа.

Поради това, че суфикните дървета представляват инструмент за търсене на текстови низове, който позволява бързо откриване на близки и еднакви последователности, SHREC представлява интерес както при сравнение на резултатите, така и при потенциални бъдещи разширения на този труд с използването на суфиксни дървета.

С помощта на CUDA и GPGPU. Съществуват разработки, които използват графичните процесори на видеокарти (general purpose computing on graphics processing units – GPGPU, GPGP, GP²U) за изчисляване на оценка на качеството и корекция на грешките. Една използвана технология е CUDA на NVIDIA [136]. По същия начин, както при SHREC, насоката на тези разработки е към секвенирането на нови геноми на единични организми, и не е директно приложима в метагеномиката.

Освен това основната им цел е ускоряване на процеса на корекция на прочити, за да се обработят бързо милиони прочити с цел осигуряване на по-голямо покритие по-бързо. В разглежданите в този труд метагеномни изследвания, изискващи най-голям изчислителен ресурс се явиха последващи операции като множественото съпоставяне построяването на филогенетично дърво, а обработката на милиони прочити не е по възможностите на използваните тук средства.

Отстраняване на прочити. Различни изследователи се спират на отстраняването на прочити като решение на проблема с грешките. Това могат да бъдат както прочити, за които е установена голяма вероятност за наличие на грешки, така и прочити, които се срещат един единствен път [63]. Такава практика е подкрепена от неравномерното разпределение на грешките сред прочитите, но не гарантира елиминацията на всички грешки, като в същото време намалява количеството на данни, върху които могат да се правят изследвания.

Предложените в настоящия труд подходи едва ли ще елиминират напълно тази практика, но предлагането на един нов критерий с по-висока точност може да намали относителния дял на отстранени данни, както и дяла на неоткрити грешки. В същото време, предложеният подход за директно включване на размит индикатор на увере-

ност в етапите за обработка на данните може да избегне част от негативния ефект на грешките без нуждата от тяхното пълно отстраняване.

2.4 Съществуващи решения и алтернативни методи

2.4.1 SHREC

SHREC [134] е алгоритъм за откриване на грешки използващ обобщени суфиксни дървета.

2.4.1.1 Обобщени суфиксни дървета

Суфиксните дървета (suffix trie) са модификация на префиксните дървета (trie) и позволят търсене по суфикс на подниз в един символен низ. Те се използват за значително ускоряване на операции със символни низове и намират приложение, както в биоинформатиката [9], така и в други задачи като ускорението на търсене с регулярни изрази [7].

Обобщените суфиксни дървета се изграждат за множество от символни низове, като съдържат всички възможни суфикси на поднизове в тях. Тяхното приложение е основно в биоинформатиката [14], където се налага търсене в множество от кратки фрагменти, и позволява търсене с линейна сложност. Всички p срещания на подниз с дължина l могат да бъдат открити в $O(l + p)$ време [51]

На всяко ребро и в двата вида дървета се съпоставя един символ от разглежданите последователности, като на всеки път от корена към листото съответства на суфикс.

2.4.1.2 Реализация на SHREC

SHREC построява обобщено суфикно дърво по множеството от прочити, които ще бъдат оценявани за грешки. За да гарантира, че за всеко отделен прочит ще съществува отделен път, към всички прочити е прибавен уникален терминиращ символ.

На дъгите и върховете на дърветата са поставени тегла. За теглото на всяка дъга, както и на върха на изхода ѝ, се използва броя на листата, които излизат от тях. Това тегло е равно на броя на прочити, в които се среща низа съответстващ на пътя от

корена до дъгата включително.

SHREC разчита на статистическите очаквания за начина на разклонение на дървото – близо до корена е вероятно всеки връх да има по четири разклонения, докато далеч от корена се очаква върховете да имат само по едно продължение. Това позволява да бъде изграден алгоритъм, който търси потенциални грешки в дъгите отдалечени от корена, но чиито качества отслабват при приближаване към него.

Потенциалните грешки се оценяват на база на теглата. При откриване на възли с повече от очакваните разклонения, високите тегла подсказват, че разклонението е в следствие на мутация, докато ниските тегла подсказват, че е в следствие на грешка. Този критерий обаче отслабва, когато бъдат срещнати разклонения прекалено близко до листата.

При откриване на грешка, алгоритъмът може да я коригира, като замени погрешното разклонение с правилното, стига суфиксите им да съвпадат.

Според опитите на авторите, SHREC открива над 96% от погрешните бази и постига точност на корекцията от над 80%.

2.4.2 V-Phaser

V-Phaser[87] използва статистически подход за откриване на грешки, който разчита не само на оценка на отделните потенциални грешки, но и на двойките грешки, които се срещат едновременно. Това има за цел да засече прочити на еволюционни разделени гени по съвпаденията на мутациите в тях.

За отделните грешки са използвани множество от случайни величини с разпределение на Бернули E_{ik}, които са 1, ако има ако има грешка на позиция i в прочит k, и 0 в противен случай. Като вероятност за успех за всяка от тях е взета очакваната вероятност за грешка p_i. Тя зависи от позицията, тъй като се очаква количеството грешки да бъде различно в различните части на прочитите.

За съвпадение на грешки на определена двойка позиции се въвежда второ множество от случайни величини E_{ijk}, които са 1, когато в четене k има грешки едновременно на позиции i и j, вероятността за което е $p_i p_j$.

За да бъдат изследвани, за метода са въведени величините X_i и X_{ij} с бройката на

грешките срещани на позиция i, както и двойките грешки срещащи се едновременно на позиции i и j. Тъй като за всяка позиция (или двойка позиции) случайните величини са еднакво разпределени, при предположение за независимост сумата им има биномно разпределение. Ако покритието на позиция i е отn_i прочити, то е в сила:

$$X_i = \sum_k E_{ik} \sim B(n_i, p_i) \tag{2.29}$$

$$X_{ij} = \sum_k E_{ijk} \sim B(n_i, p_i p_j) \tag{2.30}$$

Използвани са вероятността $f_i(x)$ грешките на позиция i да надминават определена бройка x, както и вероятността $g_{ij}(x)$ бройката от съвпадение на грешки в i и j да надмине x. Тъй като е възможно покритието на позиции i и j да бъде различно, се разглежда само бройката прочити n_{ij}, които участват в покритието и на двете позиции. Така тези вероятности изглеждат по следния начин:

$$f_i(x) = \sum_{t=x}^{n_i} \binom{n_i}{t} p_i^t (1-p_i)^{n_i - t} \tag{2.31}$$

$$g_{ij}(x) = \sum_{t=x}^{n_{ij}} \binom{n_{ij}}{t} (p_i p_j)^t (1-p_i p_j)^{n_{ij} - t} \tag{2.32}$$

Върху $f_i(x)$ и $g_ij(x)$ е приложена корекция на Бонферони, която е инструмент за измерване на груповата честота на грешки при тестване на множество хипотези. С него се въвеждат праговете t_i и t_{ij}:

$$t_i = \arg\min_t \left(f_i(t) \leq \frac{\alpha}{c} \right) \tag{2.33}$$

$$t_{ij} \arg\min_t \left(g_{ij}(t) \leq \frac{\alpha}{b} \right) \tag{2.34}$$

Тук α е степен на значимост, c е бройка позиции, за които има покритие от поне едно четене, а b е бройката двойки i, j които имат общо покритие от поне едно четене.

Ако бройката открити разлики на позиция i надминава t_i, се прави извода, че поне някаква част от тях не са грешки. По същия начин, ако бройката двойки съвпадащи разлики на позиции i и j надминава t_{ij}, то поне някаква част от тях не са предизвикани

грешки, на базата на което алгоритъма се опитва да открие грешките без да премахне естествената вариация.

Според изследванията на авторите, V-Phaser открива над 97% от грешките и над 97% специфичност върху вирусни данни събрани от заразен пациент. Съществува версия V-Phaser 2 [160], която се опитва да подобри метода на откриване на грешките допълнително.

2.4.3 Coral

Coral [130] поправя грешките в прочитите, като използва честотите на срещане на базите върху специално построено тяхно съпоставяне.

Преди извършване на корекцията се генерира таблица на k-мерите в прочитите. След това те се съпоставят със собствен подход, който е предвиден да работи добре за много близки последователности. Алгоритъма извършва многократно съпоставяне по двойки с алгоритъма на Нийдълман и Вюнш спрямо случайно избрана консенсусна секвенция, която се обновява при всяко добавяне на нова секвенция към съпоставянето.

След приключване на стъпката по съпоставяне, всички секвенции, които не са били добре съпоставени, се премахват. Използват се и други критерии за премахване на прочити, които гарантират, че няма да се извършват поправки в данни, за които честотните оценки биха дали противоречив резултат.

Самото откриване на грешки разчита на серия от честотни оценки. Първата е качеството на съпоставянето, която се изчислява като най-малката бройка бази съвпадащи с консенсусната секвенция сред всички прочити. Ако тази стойност е под предварително зададен праг $1 - e$, за съответната група прочити няма да бъде извършвана корекция.

За всяка позиция се изчислява степен на поддръжка на консенсуса – съотношението на прочити, които го потвърждават, и ако това съотношение е по-малко от предварително зададен праг $t_s \in \left[\frac{1}{2}, 1\right)$, на съответната позиция няма да бъде извършвана корекция.

При така направените ограничения, във всички останали четения и позиции, всич-

ки бази, които не съвпадат с консенсуса биват заместени с база от консенсуса.

В сравненията с SHREC на авторите, Coral успява да постигне по-добри резултати като използва чувствително по-малко памет и работи чувствително по-бързо. На редица проведени експерименти постига подобрение както в бройката открити грешки, така и в числова оценка на подобрение, която измерва броя на действителни грешки.

Глава 3

Формализация и решение на проблема за анализ и класификация на данни от паралелно секвениране

3.1 Постановка на задачите

Основната задача на този дисертационен труд е разработването на нови средства за откриване на грешки за метагеномни данни от паралелно секвениране, които са по-надеждни и по-подходящи за извършване на корекция на откритите грешки от наличните.

Разработен е оригинален аналитичен подход за откриване и корекция на грешките. Той се основава на претеглено изчисляване на честотите на срещане на всяка база по колони, като за тегла са използвани локалните сходства между двойките секвенции. Те имат за цел да неутрализират отрицателните ефекти на хетерогенната структура на данните, като приближат сходните подпоследователности и отдалечат различните, като по този начин елиминират голяма част от неправилно откритите грешки в редки секвенции, обособявайки ги локално в отделна група.

Поради трудното намиране на тестово множество данни за валидация са предложени два индиректни подхода за валидация – един, основан на неколкократна корекция на грешки, която използва симулирани грешки, и един, основан на корекция на греш-

ките в голямо множество от данни и достатъчно малко негово случайно подмножество. За симулацията се изгражда статистически профил на грешките върху множество от данни с известни грешки, който е специфичен за използвания апарат за секвениране, и който съответства на разпределението на истинските грешки.

Предложен е метод за симулация на „виртуални“ грешки при конструирането на обучаващо множество от примери, което да се използва при създаване на класификационни модели за откриване на грешки, основани на изкуствени невронни мрежи и други средства на машинното самообучение като случайни гори. Неструктурираните зависимости и хаотичната структура на данните ги правят подходящи за приложение на инструменти като ИНМ, поради техните способности за правене на изводи. Когато аналитичните решения не постигат достатъчно добър резултат, апаратът на ИНМ може да бъде използван за създаване на по-прецизен класификатор на грешките, който да подобри допълнително качеството. Това би било за сметка на по-малката яснота за начина, по който те биват идентифицирани.

Вторична задача на дисертационния труд е изграждането на система за изпълнение на управляеми биоинформатични работни потоци. Разработена е софтуерна библиотека, която освен реализацията на алгоритмите за откриване и корекция на грешки, за симулация на грешки и валидация, поддържа всички необходими операции, използвани за обработка на данните. Това включва множествено съпоставяне на секвенции, клъстериране на секвенции. Разработена е и система, която позволява тези операции да бъдат включени в описания на работни потоци.

Проведени са експерименти, които показват, че използването на локалната честота на срещане за елиминиране на излишните корекции води до резултати, които отговарят на очакванията. Постигат се по-малък брой корекции без пропускане на допълнителни симулирани грешки, което е очаквано при подобряване на качеството на класификация. Аналогичен резултат се наблюдава и след допълването на аналитичния подход с класификатор построен с методите на машинното самообучение. Тази зависимост се запазва и когато средствата от този труд са сравнени с алтернативни инструменти като SHREC. Предложените подходи водят до постепенно подобряване на качеството, а тяхната комбинация – до качество по-добро от постигнатото с SHREC.

Това означава, че предложените подходи водят до поетапно подобрение в качест-

вото на премахване на шум.

3.2 Входни данни

За метагеномни изследвания често се използва 16S РНК. Тя съдържа силно консервирани региони, които са идентични или сходни измежду много голяма бройка микроорганизми, което дава много добра отправна точка при извършване на междувидови сравнения. В същото време съдържа и хипервариабилни (силно вариращи, hypervariable) участъци. Те могат да бъдат използвани за откриване на разликите между секвенциите при клъстериране на групи, отговарящи на отделните видове микроорганизми, както и при търсенето на еволюционните връзки между тях. Това прави 16S РНК много подходяща за изследване на биоразнообразието и при построяването на филогенетични дървета [150].

За целите на този труд са използвани примерни множества от данни, съдържащи прочити с дължина между 300 и 500 бази. Те са разделени в проби, съдържащи между 30000 и 50000 секвенции, след извършване на предварително филтриране (от тях са отстранени прочитите с непрочетени позиции, както и прекалено късите и прекалено дългите прочити, за да се елиминират проблеми, свързани със статистически отклонения). Пробите са секвенирани с 454 платформата [146] на Roche, която е подходяща за метагеномни експерименти поради това, че генерира по-дълги прочити от платформите, използвани при секвениране на геноми на отделни организми.

```
TCTCTATGCGCCATTGTAGCACGTGTGTA…GGTAAGGTTCTTCGCGTAGCTTCGAATTAAACC
TCTCTATGCGCCATTGTAGCACGTGTGTA…GTAGGTAAGGTTCTTCGCGTTGCATCGAATTAA
TCTCTATGCGCCATTGTAGCACGTGTGTA…GGTAAGGTTCTTCACGTTGCTTCGAATTAAACC
TCTCTATGCGCCATTGTAGCACGTGTGTA…GTAAGGTTCTTCGCGTAGCTTCGAATTAAACCA
TCTCTATGCGCCATTGTAGCACGTGTGTA…TAAGGTTCTTCGCGTAGCTTCGAATTAAACCAC
TCTCTATGCGCCATTGTAGCACGTGTGTA…GGTAAGGTTCTTCGCGTTGCTTCGAATTAAACC
```

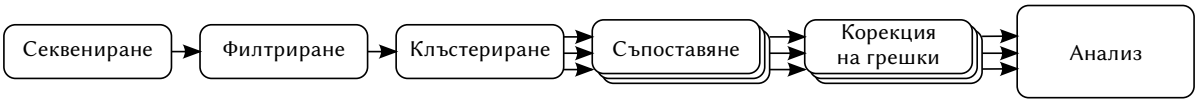
Фигура 3.1: Откъс от входните данни

Отрязък от входните данни може да се види на фигура 3.1. Показани са началото и края на няколко отделни прочита, които са били секвенирани от пробата. Възможно е те да са дошли от един и същи организъм в нея, както и да са дошли от различни, но много сходни организми. Отделните редове в данните представляват отделни прочи-

ти или отделни видове, докато колоните представляват отделните позиции, в които, след извършване на множествено съпоставяне, се очаква да има съответствие между прочитите и видовете.

3.3 Предварителна обработка

За да могат да бъдат приложени разработените в този труд подходи за корекция на грешките, данните подлежат на няколко предварителни обработки, както е илюстрирано на фигура 3.2.



Фигура 3.2: Анализ на данни от метагеномно секвениране

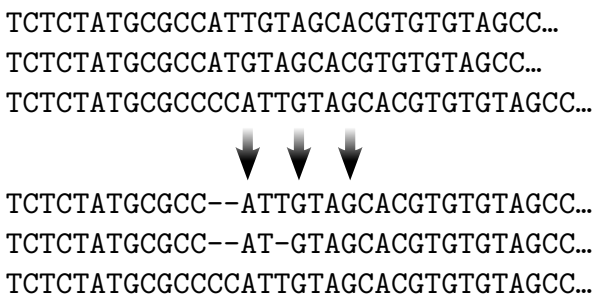
Първоначално върху данните се извършва филтриране. В нашия случай при него бяха премахнати всички прочити, чиято дължина е под 300 бази, както и всички прочити, които съдържат позиции с неизвестни бази. Останалите прочити бяха подрязани до дължина от 500 бази. Това осигурява по-добро качество на използвани данни, тъй като е премахната опашката на секвенциите, където има значително по-висока честота на грешки [47]. В същото време това премахва и статистически отклонения предизвикани от прекалено голямо вариране в дължините на прочитите.

След филтрирането е необходимо секвенциите да бъдат клъстерирани, което в нашия случай е извършено с помощта на готово софтуерно решение – CD-HIT [80]. Клъстерирането осигурява работа с подмножества, в които не се съдържат прекалено далечни секвенции. Тъй като всички предложени методи за корекция на грешки разчитат на съпоставени прочити, върху всеки клъстър е изпълнена процедура на съпоставяне по описания по-долу комбиниран подход.

Множествено съпоставяне

Множественото съпоставянето на секвенции по някой от описаните в 2.3.1.3 и 2.3.1.4 методи е най-значимия елемент от предварителната обработка на данните. На него тук

е обърнато специално внимание, тъй като за извършването му не беше достатъчно да бъде използвано готово софтуерно решение.



Фигура 3.3: Необходимост от съпоставяне

На фигура 3.3 е показан пример, който демонстрира необходимостта от извършване на такава процедура. Както се вижда, във втората последователност база на позиция 15 е изпусната (тоест в прочита има делеция). Това може да е в следствие на мутация в единия от двата индивида, или в следствие на грешка при четене. И в двата случая, разместването, предизвикано от делецията, прави невъзможно извършването на каквито и да било сравнения по колони, понеже всички бази след 15-та колона не си съответстват вертикално, освен ако тези разминавания не бъдат елиминирани с помощта на съпоставяне.

Един подход за множествено съпоставяне е описан в 2.3.1.3. Той поетапно съпоставя двойките секвенции, разположени по водещо дърво, получено след йерархично клъстериране. Аналогия на този подход е използвана тук за подобряване на производителността на съпоставяне с използване на готови софтуерни инструменти.

При проведените експерименти бе установено, че дори готовите софтуерни решения предвидени за големи множества данни, не се справят достатъчно добре с целите масиви от входни данни за приемливо време, или не водят до достатъчно добро качество на резултатите. Някои софтуерни решения дори отказаха да приемат всички секвенции. Трябва да се отбележи, че и резултата от директна реализация на алгоритъм 2 от 2.3.1.3 не доведе до задоволителен резултат, защото в него отсъстват всички подобрения въведени с итеративните алгоритми описани в 2.3.1.4.

За съпоставяне на цялото множество беше използван следният подход. Беше приложено грубо клъстериране на данните отново с помощта на CD-HIT [80]. Беше из-

вършено множествено съпоставяне във всеки клъстер с готов софтуерен инструмент – MAFFT [67] или MUSCLE [38]). След това беше извършено съпоставяне между двойките клъстери по подобие на на стъпки 3 и 4 от алгоритъм 2. MAFFT има и вградена функция за съпоставяне на двойки клъстери, с която последната стъпка може да се улесни.

При използване на тази процедура, производителността беше повишена значително, а в същото време качеството беше значително по-добро от изпълнение на MAFFT или MUSCLE в режим за бързо изчисление.

Така беше установено, че макар и съпоставянето да е решима задача, готовите софтуерни решения не могат да се използват направо, без ръчно комбиниране с решения за клъстериране. Това е проблем, с който се среща всеки изследовател занимаващ се с метагеномика, и показва ясно необходимостта от създаване на софтуер за изпълнение на управляеми работни потоци, който позволява тяхното лесно споделяне.

3.4 Разработени аналитични методи

3.4.1 Аналитичен подход за корекция на грешки

Грешките в прочитите са основен проблем при метагеномните изследвания и практиката да се отстраняват всички прочити, в които има съмнения за грешки, не е оптимално решение на този проблем. За съжаление грешките (заместванията, инсерции и делеции) са смесени с мутации от същите видове и различаването им не е лека задача.

Мутациите имат голямо значение за еволюционните изследвания и носят значима информация за развитието и размножаването на видовете микроорганизми. За разлика от грешките при секвениране, които са разпределени случайно в пробите, мутациите претърпяват различно развитие според това дали са били полезни за развитието на организма. Полезните се унаследяват и размножават, докато вредните убиват организма и прекъсват еволюцията му, поради което не се очаква те да имат същия дял на участие в секвенционните данни. Това от една страна прави оцелелите мутации интересни с информацията, която те носят, а от друга страна представя една възможност за класификацията на двете категории разлики. Мутациите, които са били наследени,

ще се отличават по своите измерими характеристики като честота на срещане.

За работа с потенциалните грешки е въведена степен на увереност (confidence score), която може да бъде изчислена с различни функции $s : R \times \mathbb{N} \to [0, 1]$. Стойността $s(r, k)$ е степента на увереност във верността на базата r_k, която се намира на позиция k в прочита r от множеството на прочити R. Тези оценки са използвани както за откриване и корекция на грешките чрез сравняването им с праг T^e (в 3.4.1.4), така и директно като размит индикатор на увереност в алгоритмите за обработка на геномни данни (в 3.5).

Популярен инструмент за откриване на грешките е честотата на срещане. При секвенирането на генома на един организъм, това води до достатъчно добър резултат, понеже всички прочити са от един и същ организъм, и се очаква, че всички прочити от един и същи регион са идентични, и че една мутация ще се среща във всеки прочит.

3.4.1.1 Наивен подход за откриване на грешките

Ще наречем директното използване на честотата на срещане „наивен подход“. Най-лесно е да пресметнем честотата на срещане на различните нуклеотидни бази във всяка колона, и тези, които се срещат по-рядко от някакъв предварително зададен праг, да се считат за грешки.

Предположението е, че дори в една метагеномна проба, един индивид ще се среща повече от един път. Така макар и да са по-редки от грешките, оцелелите мутации ще се срещат много по-често в множеството от данни, понеже са били наследени след размножаване или понеже индивидът в мутацията се среща достатъчно често в пробата. Това предположение не е достатъчно силно, но дава добра основа, върху която да се надгради подобрен подход.

Честотата на базите по колона може да се опише със следната „наивна“ функция за степен на увереност (confidence score). С нея може да се изчисли честотата на срещане на база r_k от прочита r в колона k сред множеството от прочити R.

$$s_{\text{simple}}(r, k) = \sum_{p \in R}^{p \neq r} \frac{[r_k = p_k]}{n - 1} = \frac{\sum_{p \in R}^{p \neq r} [r_k = p_k]}{\sum_{p \in R}^{p \neq r} 1} \tag{3.1}$$

3.4.1.2 Подход за откриване на грешки, основан на локално сходство

Основният недостатък на използване на честотата на срещане е, че не взема предвид последователности, които се срещат рядко. Например, ако даден индивид е попаднал само един път в пробата, s_{simple} ще оцени много бази от него с оценка под прага на грешка. А дори и дадена рядка последователност да се среща по-често от прага на грешките, оценката на грешки сред нейните секвенции ще трябва да се осъществява при друг праг.

За да се вземе предвид хетерогенното съдържание на множествата от данни, са предложени няколко подобрения, които използват локалната степен на сходство между двойките секвенции при пресмятане на честотата на срещане. Подобреният подход трябва да отговаря на следните изисквания (i):

- Несъответствие между два сходни прочита трябва да е по-важно от несъответствие между два различни прочита. По-общо, важността на едно несъответствие между двойка прочити трябва да бъде пропорционална на сходството между тях.
- Сходството около несъвпадението трябва да има по-голямо значение от сходството в по-отдалечените региони на секвенциите. По-общо, важността на сходството в дадена позиция трябва да е обратно пропорционална на разстоянието от оценяваното съвпадение или несъвпадение.

```
TCTCTATGCGCC ATTGT AGCACGTGTGTAGCC… (6716)
TCTCTATGCGCC ATAGT AGCACGTGTGTAGCC… (20)    <- p
TCTCTATGCGCC TCACG AGCACGTGTGTAGCC… (20)    <- r
TCTCTATGCGCC TCTCG AGCACGTGTGTAGCC… (1)
             i k
```

Фигура 3.4: Сходство в прозорец около оценяваната позиция

Причината да се вземат предвид само разликите в близки видове е, че вероятността да се срещне последователност от грешки е експоненциално по-малка от вероятността за единична грешка. Така, ако цял регион е променен, е много по-вероятно това да е отрязък от рядко срещан организъм, отколкото да е последователност от грешки.

Освен това различните секвенции ще се отделят в различни групи и грешките в тях ще се оценяват независимо една от друга.

Алгоритъм 5. Откриване на грешки, основано на локално сходство

1. Около всяка оценявана позиция се изгражда прозорец, в който ще бъде ограничено сравняването на последователностите.

2. За всяка двойка прочити, в които се търси несъвпадение в съответната позиция, се изчислява сходството между тях в прозореца. При оценката на сходство, позициите, по-близки до края на прозорецау се оценяват с по-ниски тегла от тези близки, до оценяваната позиция.

3. При съвпадение степента на доверие за всяка от двете секвенции се увеличава със степента на сходство, разделена на сумата на всички степени на сходство за съответния прочит.

Формулата (3.1), разширена с оценка за сходство, придобива следния вид. За региони, които имат сходство 1, резултатът ще е същият като със „наивната“ формула, докато региони със сходство 0 ще бъдат пренебрегнати при пресмятането на сходството.

$$s_{\text{weighted}}(r,k) = \frac{\sum_{p\in R}^{p\neq r} \text{sim}(r,p,k)[r_k = p_k]}{\sum_{p\in R}^{p\neq r} \text{sim}(r,p,k)} \tag{3.2}$$

Изборът на добра sim функция е труден. Първото разгледано предложение е за експоненциално намаляваща функция. При използването на такава, разликите на определено отстояние от оценяваната позиция престават да имат значение, тъй като стойността им бързо клони към 0, и размерът на прозореца престава да бъде съществен параметър.

Нека q е някакъв параметър, който подлежи на оценка. Прозорецът около позиция k за прочит r и сходството между прочит r и k в прозореца около позиция k ще се изчисляват по следния начин.

$$\text{window}(r,k) = \{i : \exists r_i, i \in [k-w, k-1] \cup [k+1, k+w]\} \tag{3.3}$$

$$\text{sim}(r,p,k) = \frac{\sum_{i \in \text{window}(r,k)} q^{|k-i|}[r_i = p_i]}{\sum_{i \in \text{window}(r,k)} q^{|k-i|}} \tag{3.4}$$

Сложността на представения алгоритъм е $O(wn^2)$. В същото време обаче е възможно да бъде изграден кеш на всички различни последователности, които се срещат в отделните прозорци, което може да доведе до подобрение за множества от данни, в които има много сходни прочити – каквито често са геномните проби, особено след клъстериране. След използване на кеш, сложността на алгоритъма би била $O(\min(ne^w, wn^2))$. Изглежда, че експоненциалната сложност спрямо размера на прозореца прави подобна оптимизация излишна, но в действителност броя операции за всеки един прозорец се определя от реалното разнообразието вътре в него, което никога не съдържа всичките 5^w последователности[1].

Трябва също да отчетем любопитен страничен ефект от използването на този подход, основан на локалното сходство. Обикновено при секвениране, за оценка на качеството секвенциите на един организъм се използват само прочити от същия организъм. За съжаление в метагеномиката това не е напълно възможно и данните винаги ще съдържат прочити от различни организми, които внасят допълнителна неяснота. Оценката за сходство, от една страна елиминира регионите, в които тези секвенции са напълно различни, но от друга страна запазва регионите, които са били запазени дори в двойки различни биологически видове, което дори оставя възможността информацията от единия вид да потвърди информацията за другия.

3.4.1.3 Остра тегловата функция за локално сходство

Ползата от приложените средства за справяне с грешките, били те корекция с праг или използването на размит индикатор, силно зависи от качествата на избраната функция на увереност s. Например, при прилагането на s_{simple} за целите на размито йерархично клъстериране (според 3.5), резултатите очаквано се влошиха значително, защото дълги заместени региони бяха пренебрегнати от оценката за разстояние. Въведено в

[1] Възможните символи са 4-те бази и специалния символ за интервал

s_{weighted} тегло противодейства на този отрицателен ефект, като в същото време успява да заглуши единичните грешки, но все още има възможност за подобрение, като изборът тегловата функция за локално разстояние $\text{sim}(r, p, k)$ е ключов.

Един основен недостатък на локалното разстояние от (3.4) е, че внася допълнителна неяснота по границите на въпросните заместени региони. Освен това при използването му, размитото йерархично клъстериране представено по-късно в 3.5 е прекалено толерантно към промени в двойки, тройки и кратки фрагменти, въпреки че подобни рядко са причинени от натрупване на грешки.

По тези причини беше въведена и алтернативно тегло за локалното разстояние, което има по-„остър“ ефект, нарастващ експоненциално с всяка разлика. Нека q е параметър, който подлежи на оценка, и $0 < q << 1$. Тогава за алтернативно тегло може да се използва и следната функция, която ще наричаме *остра*.

$$\text{sim}(r, p, k) = \prod_{i \in \text{window}(r,k)}^{r_i \neq p_i} q^{\frac{1}{|k-i|}} \tag{3.5}$$

Експоненциалното нарастване на разстоянието при разлики отчасти намалява ефекта последователности от промени, които са с малка вероятност да бъдат грешки, като запазва преимуществата на (3.4) и продължава да отговаря на изискванията (i) (от страница 70). Практически при проведените експерименти положителният ефект от това алтернативно разстояние не беше достатъчно убедителен, за да бъде избрано като основно пред (3.4).

3.4.1.4 От откриване към корекция на грешките

Формула (3.1) и (3.2) дават числова оценка за степента на увереност, че базата, заемаща определена позиция в секвенцията, е прочетена правилно. За някои определени задачи (като разгледаните в 3.5) подобна оценка може да бъде използвана наготово, но за да се разграничат случаите на предполагаеми грешки и правилно прочетени данни, оценките трябва да се сравнят с някакъв числов праг. В най-опростения си вид това би било единствено число T^e, под което увереността ще се счита за недостатъчна.

В по-обобщени случаи прагът би могъл да зависи от оценяваната база ($T^e(r_k)$), от позицията ѝ в оценяваните прочити (T^e_k или $T^e_{r,k}$), от външна информация за тях като

оценката за качество на секвенииращия апарат ($T^e(Q_{r,k})$). В някои задачи би било полезно или необходимо използването на повече от един праг – като праг на грешка, праг на несигурност, праг на точкова мутация (T^e, T^u, T^{snp}).

Използването на сложен променлив праг, който зависи от позицията или от данните за нея, би могло по-точно да разграничи потенциалните грешки, но това би било за сметка на по-големия брой параметри, които ще трябва да бъдат оценени. При ограничените възможности за валидация, използването на променлив праг или на множество прагове може да се окаже контрапродуктивно, поради което фокуса на тази работа е върху избора на единствен прост числов праг. Начини за неговия избор са разгледани в 3.4.2.

При избран праг $T^e_{r,k}$, и избрана функция на увереност $s(r,k)$ (например $s(r,k) = s_{\text{weighted}}(r,k)$), за грешка ще се считат позиции, които следното условие за грешка.

$$s(r,k) < T^e_{r,k} \tag{3.6}$$

За да бъдат приложени подходите от 3.4.1.1 и 3.4.1.2 за корекция на грешки, е необходимо те да бъдат разширени. Един възможен начин е да се приложи функцията $s(r,k)$ от (3.1) и (3.2) за всички възможни кандидати, с които могат да бъдат заместени базите r_k, за които е в сила условието (3.6).

Нека означим с $r|^{\mathsf{X}}_k$ прочита r, в който позиция k е била заменена с базата X. Тогава, за заместване на база r_k, която удовлетворява условието за грешка (3.6), ще бъде използвана базата от следното правило за заместване.

$$\underset{\mathsf{X}\in\{\mathsf{A},\mathsf{C},\mathsf{T},\mathsf{G},\varnothing\}}{\arg\max}\ s(r|^{\mathsf{X}}_k, k) \tag{3.7}$$

Подобен подход към заместването предполага, че е използван числов праг $T^e_{r_k}$, който не зависи от съдържанието на оценяваната позиция. Използването на по-гъвкав праг е несъвместимо с предложеното правило (3.7), освен ако самият праг $T^e(r_k)$ не се включи в правилото. Това може да бъде извършено по различни начини, които зависят от избраната функция s и отношението ѝ към прага, като например:

$$\underset{\mathsf{X}\in\{\mathsf{A},\mathsf{C},\mathsf{T},\mathsf{G},\varnothing\}}{\arg\max}\ s(r|^{\mathsf{X}}_{k},k)-T^{e}(\mathsf{X}) \tag{3.8}$$

$$\underset{\mathsf{X}\in\{\mathsf{A},\mathsf{C},\mathsf{T},\mathsf{G},\varnothing\}}{\arg\max}\ \frac{s(r|^{\mathsf{X}}_{k},k)}{T^{e}(\mathsf{X})} \tag{3.9}$$

За да се избегне излишно усложнение, използването на сложни правила като (3.8) и (3.9) в този труд е избегнато. При нужда от различни оценки на различните бази се разчита на промени във функцията s, които водят до по-голяма гъвкавост. Обичайна практика е за различните класове несъвпадения да се използва различно тегло [145] вместо използваните в (3.1) и (3.2) стойности 1 и 0 за съвпадение и несъвпадение. Те най-често са представени като матрица на заместване. Такива примери са разгледани в 3.5.

Забележка 1. За целите на задачата интервалите, вмъкнати при съпоставянето на секвенциите от 3.3, означени с $\varnothing$ (и с „-“ в компютърното представяне) се разглеждат като пълноправна база, макар и да са технически инструмент без биологичен смисъл. При необходимост тяхното разграничаване може да стане числово чрез добавяне на допълнителни тегла в $s(r,k)$, следвайки матриците на заместване.

3.4.2 Индиректни подходи за валидация

За да е възможно използването на предложените в 3.4.1 подходи за откриване и корекция на грешки е необходим приложим и обективен подход за валидация. Валидацията е необходима както да се оцени подобрението от прилагане на метода на сходството от 3.4.1.2 и качеството от прилагането му, така и за избор на подходящ праг T^e за (3.6), при който откриването и корекцията на грешки работят най-добре. Освен това, възможността за валидация е ключова за разработването на подход за откриване на грешки, основан на методи на изкуствения интелект, който е засегнат в предложения в 3.6 подход за обучение на невронни мрежи.

За съжаление съществуват големи практическите трудности за намиране на надеждни тестови множества, които да са представителни за данните в една секвенирана метагеномна проба. Достъпните множества без грешки са били получени след прилагане на някаква форма на филтриране на погрешните прочити и рядко ще имат вида,

който имат суровите данни – обект на разглежданата задача. При данните, извлечени от бази от данни, също съществува рискът част от редките мутации да не са представени – а точно тяхното запазване би било сред целите на един подобрен подход за откриване на грешки.

В следствие на това в тази работа са предложени два подхода за индиректна валидация на корекцията на грешки, които ще бъдат използвани за оценяване на представените методи.

3.4.2.1 Валидация чрез множество и подмножество

Основна пречка при валидацията на корекцията на грешки е невъзможността една проба да бъде взета отново в същия си вид, тъй като всяка взета проба е различна. Ако беше възможно да се повтори, валидацията на данните щеше да може да бъде извършена чрез многократно потвърждаване на генетичния код на всички микроорганизми, които се намират в изследваната среда. Достъпните пълни поправени секвенции щяха да бъдат достатъчни, за да бъде оценен всеки резултат, получен след извършване на корекция на грешки. Също така, ако цената на такъв процес беше достатъчно ниска, нуждата от такава корекция на грешки нямаше да е толкова голяма.

Друг начин, по който това може да се изкаже, е, че ако може да се събере достатъчно голямо покритие от прочити на търсените генетични данни, то тяхното съдържание ще бъде потвърдено с достатъчно голяма степен на достоверност – за това хипотетично би било достатъчно да се събере множество от данни (ii), в което всяка генетична подпоследователност се среща многократно. Но това също така означава, че колкото по-голямо покритие е на разположение, толкова по-голяма е достоверността данните – тези със значително по-голямо са значително по-достоверни от тези със значително по-малко. А при корекция на грешките, използваща метод, зависещ от степента на покритие, корекцията приложена върху по-голямото ще има много по-добра успеваемост.

Това позволява данните от едно голямо, но идентично по произход и съдържание, множество да бъдат използвани за оценка на качеството на едно значително по-малко такова. По тези съображения, за валидация на подход на корекция на грешките беше разработена следната процедура.

Процедура 1. Процедура на подмножествата

1. Избира се най-голямото достъпно множество от метагеномни данни прочетени наведнъж.

2. Извършва се корекция на грешките. Полученият резултат ще бъде използван като стандарт за валидация.

3. Избира се случайно подмножество, което е значително по-малко от цялото, например, десетократно или стократно.

4. Извършва се корекция на грешките върху подмножеството и се сравнява със стандарта, получен в 2.

Както цялото множество, така и неговото подмножество, са представителни за данните, извлечени от едно изпълнение на апарата за геномно секвениране. И двете имат едно и също разпределение на грешките, които трябва да бъдат коригирани, така и на мутациите, които трябва да бъдат запазени. И двете са изградени от данни с еднакви характеристики, включително дължина, съдържание, разпределение на видовете бактерии и прочее. В същото време, цялото множество осигурява значително по-голямо покритие, което осигурява много по-голяма степен на достоверност на данните, и повишава качеството на корекцията на грешки над това, което има същата на подмножеството.

Ако цялото множество беше достатъчно голямо, тоест ако имаше размера на недостижимото множество (ii) (от страница 76), то щеше да бъде достатъчно за пълна валидация на корекцията на грешки. Но дори когато това не е изпълнено, разликата между постигнатите резултати може да бъде използвана, за да се направи сравнителна оценка между два алгоритъма, като се оцени кой прави повече излишни корекции и пропуска повече грешки при данни с много по-ниска достоверност.

Основният недостатък на този подход е, че дори при слаби изисквания за размера на множеството и съотношението между него и подмножеството, той изисква много голям обем от данни, които не са лесно достъпни за изследване. По тази причина за валидация бе прилаган основно подходът, представен в 3.4.2.3.

3.4.2.2 Симулация на грешки

Съществуват различни решения за симулация на процеса на секвениране, които генерират данни, тръгвайки от известни секвенции [59, 120]. Макар и да представляват интересен инструмент за валидация, на подобни средства се гледа с недоверие поради това, че е трудно да се оцени достоверността на подобна симулация. Освен това подобни решения за симулация са специфични за използвания апарат за секвениране и надграждането им за други модели изисква спецификации, които не винаги са публично достъпни. Не беше открито готово решение симулиращо резултатите получени от същия Roche апарат използван за секвениране на тестови данни за този труд (виж 3.2).

От друга страна, за симулация само на грешките в данните са достатъчни статистически характеристики на тяхното разпределение. В частност, би било достатъчно да се знае честотата на грешки на определено разстояние от началото на прочитите. Това е така, понеже при процесът на секвениране базите се четат последователно [124], и макар и да има други фактори, които оказват влияние на вероятността за грешка [47], разстоянието от началото е най-значим.

Беше разработен симулатор, който използва тези честоти. Многократно бяха секвенирани проби от известна повтаряща се последователност, за която наивният подход от 3.4.1.1 е достатъчно добър. След съпоставяне и корекция на грешките бяха изградени консенсусни последователности, които бяха използвани за еталон за оценка на всички прочити. За всяка колона k бяха измерени честотите на заместванията N_k^s, честотите на инсерции N_k^i и честотите на делеции N_k^d. Беше разработен софтуерен генератор на грешки, който използвайки тези честоти, за да внесе случайни нови грешки в множество от ДНК или РНК секвенции.

Нека означим с τ_k^r позицията, на която се намира оригиналната база r_k след съпоставяне, така че r_k да е позицията преди съпоставяне, а $r_{\tau_k^r}$ е позицията след съпоставяне. Нека също така означим консенсусната секвенция с r^*. Тогава честотите на грешки се пресмятат по следния начин.

$$N_k^s = \sum_{r \in R} \frac{[r_{\tau_k^r} \neq r^*_{\tau_k^r} \wedge r^*_{\tau_k^r} \neq \varnothing]}{|R|} \tag{3.10}$$

$$N_k^i = \sum_{r \in R} \frac{[r^*_{\tau_k^r} = \varnothing]}{|R|} \tag{3.11}$$

$$N_k^d = \sum_{r \in R} \frac{[\tau_k^r - \tau_{k-1}^r > 1]}{|R|} \tag{3.12}$$

Нека χ да генерира случайно реално число при всяко свое упоменаване, и нека $0 < \chi < 1$. Симулацията на грешките се извършва по следната процедура.

Процедура 2. Симулация на грешки

- За всеки прочит $r \in R$:
  - За всяка позиция k, за която съществува r_k:
    - Ако е в сила $\chi < N_k^s$, то r_k се заменя с произволна база от `A`, `C`, `T` или `G`, различна от r_k.
    - Ако е в сила $\chi < N_k^i$, то след r_k се вмъква произволна база от `A`, `C`, `T` или `G`, която след приключване на обработката на r ще заеме позиция r_k.
    - Ако е в сила $\chi < N_k^d$, базата r_k се премахва и след приключване на обработката на r позиция k няма да съществува.

Така разработената процедура за симулация би могла да се разшири да взема предвид дължините на инсерциите и делециите, както и да разглежда специалния случай, при който секвениращият апарат удължава и скъсява мононуклеотидни повтори (`AAAAA....`). Тъй като обаче разработваните подобрения, основани на корекция на грешки, на този етап на разработка не вземат предвид зависимостите вътре в един прочит, това не беше сметнато за необходимо.

Един от основните недостатъци на подобна симулация е, че отново изисква голям обем от данни за изследване, за да станат оценките за честотите на грешки достатъчно достоверни. За щастие това е по-лесно за осъществяване, отколкото намирането на данни за подхода на подмножествата, понеже може да се използват резултатите от много проби и много изпълнения на процедурата по секвениране.

3.4.2.3 Валидация чрез неколкократна корекция на грешки

На базата на симулираните грешки е предложен индиректен подход за сравнение на инструменти за корекция на грешките, както и за сравнение на различни стойности за прага на грешка. Целта да се намери *по-добър* подход отговаря на целта да се *намали броя на нежелани корекции при запазване на броя пропуснати грешки* (iii). При естественото предположение, че увеличаване на прага е достатъчно, за да се намали броя на пропуснати грешки, то намаляване на броя на нежелани корекции е достатъчно, за да очакваме, че подходът е бил подобрен.

За да използваме симулираните грешки, за да оценим дали има такова подобрение, е разработена следната процедура при избран подход за корекция на грешките.

Процедура 3. Валидация чрез неколкократна корекция на грешки

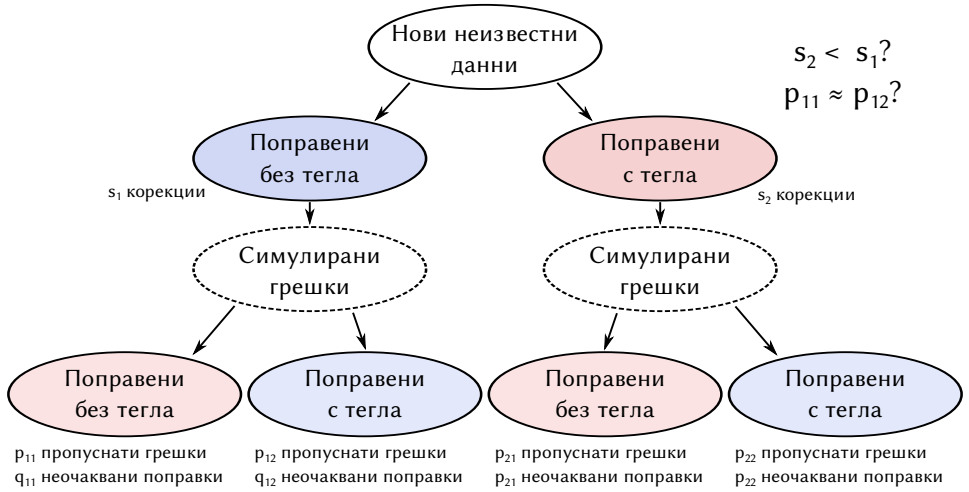
1. Вземат се изходните данни $0e$ от апарата за секвениране.
2. Прилага се подход за корекция на грешките върху $0e$, и се получават поправените данни 1.
3. Използвайки процедура 2 се симулират грешки в данните 1 чрез, и се получават данните $1e$.
4. Прилага се подход за корекция на грешките върху $1e$ и се получават поправените данни 2.
5. Оценяват се извършените корекции – взима се техния брой, както и броя на пропуснати симулирани грешки при корекцията на $1e$ към 2.

За по-голяма яснота, поредицата от данни, получени след изпълнение на тази поредица от стъпки, са показани в таблица 3.1 и са визуализирани във фигура 3.5

Въпреки че внасянето на нови грешки в множество от данни, което вече е съдържало такива, може да изглежда странно, тази процедура води дава няколко полезни числови оценки. Ако профилът на грешките, симулирани с процедура 2, отговаря на действителността, то сравненията между 1, $1e$ и 2 са достатъчни, за да се направи оценка на пропуснатите грешки и разликата в броя на корекциите.

	поправени	с грешки
Генерация 0	организъм	0e
Генерация 1	1	1e
Генерация 2	2	

Таблица 3.1: Поредица от множества с данни при неколкократна корекция



Фигура 3.5: Валидация чрез симулация на грешки и двукратна корекция

Тъй като грешките са достатъчно редки, броят на пропуснати симулирани грешки трябва да е приблизително същия като броя пропуснати действителни грешки. Така, ако двата подхода покажат еднаква успеваемост за откриване на симулирани грешки при стъпка 4 ($1e \to 2$), то може се счита, че те откриват еднакво добре и действителните грешки. Ако това е така, то при стъпка 2 ($0e \to 1$), където самите грешки са неизвестни, би било достатъчно да се оцени броя корекции. Подходът, който извършва по-малко такива при същия брой открити грешки, отговаря на изискването (iii) (от страница 80), което го оценява като *по-добър*, тъй като броят на нежелани корекции остава да е намалял.

Може да се направят и някои по-силни заключения, като това, че броят на нежелани корекции не само е намалял, но може да се пресметне като от общия брой на корекции, извършени на стъпка 2 се извади броя открити грешки на стъпка 4. Няма обаче никакви гаранции, че броят на откритите грешки на двете стъпки е еднакъв – очакваме единствено, че той е равен за двата подхода на всяка отделна стъпка. Все пак,

ако направим това по-силно предположение, бихме могли да използваме тази процедура и при избор на праг, като се опитаме да изравним броя на пропуснатите грешки с броя нежелани корекции.

Забележка 2. За пълно изследване процедура 3 не се изпълнява с един и същ подход за стъпки 2 и 4, а се изпълняват с всички възможни двойки подходи. Ако търсеното условие е изпълнено независимо от използвания подход в стъпка 2, това ни дава много по-голяма увереност в резултата.

3.5 Размит индикатор на увереност в анализа на метагеномни данни

Много от алгоритмите, прилагани за обработка на данни от паралелно секвениране, както в метагеномиката, така и в други геномни изследвания, разчитат на използването на матрици на разстоянието. В разглежданите задачи най-често става дума за разстоянията между двойките секвенции в множеството от данни. Те се използват например за йерархично клъстериране, чрез което да се построи филогенетично дърво, което описва биоразнообразието и еволюционните връзки между видовете. Подобно приложение на матрици на разстоянието и йерархично клъстериране се използва и за множественото съпоставяне на секвенциите, разгледано в 3.3.

Когато една процедура разчита само на разстоянията между двойките секвенции, тя лесно може да се модифицира, за да използва размитият индикатор на увереност $s(r, k)$ като се модифицира съответната функция за разстояние. Подобна директна промяна е възможна и за всички процедури, които разчитат на индикация за съвпадения и несъвпадения, макар че при тях може да има нужда от по-специфичен подход.

Често използвана функция за разстояния между секвенции е разстоянието на Хеминг [114], което измерва частта от секвенциите, която се различава. Ако вземем секвенциите r и p, които са с еднаква дължина m, то разстоянието на Хеминг се изчислява по следната формула.

$$H(r,p) = \sum_{i=1}^{m} \frac{[r_i \neq p_i]}{m} = \frac{\sum_i [r_i \neq p_i]}{\sum_i 1} \tag{3.13}$$

Забележка 3. Най-често се борави със секвенции с различна дължина. Тъй като обаче в общия случай тези секвенции представляват част от някаква по-дълга секвенция, може да пренебрегнем всякаква част, която лежи извън общата им дължина. Това може да доведе до някои отклонения, когато дължините на секвенциите варират много, което е част от причината да съществува практика секвенциите, чиято дължина е извън определен интервал, да се отстраняват, а останалите да се подрежат до равна дължина.

За да получим смислен резултат при приложението на разстояние на Хеминг, секвенциите трябва да минали през съпоставяне, тъй като регионите сравнявани за съвпадения трябва да си съответстват. Преди такова съпоставяне да бъде налично, например при изчислението на самото съпоставяне, може да бъде използван друг вид разстояние, като например модифицирано разстояние на Левенщайн [170]. Тъй като взема предвид инсерциите и делециите, то може да бъде използвано и преди да е било извършено съпоставяне. Негова близка алтернатива е да се извърши съпоставяне по двойки, върху които да се пресметне разстоянието на Хеминг.

За секвенции r и p с дължини m_r и m_p, разстоянието на Левенщайн (преди нормализация) ще бъде равно на $L_{r,p}(m_r, m_p)$, където L е дефинирано по следния начин:

$$L_{r,p}(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ L^{sbs}_{r,p}(i,j) & \text{otherwise} \end{cases} \tag{3.14}$$

$$L^{sbs}_{r,p}(i,j) = \min \begin{cases} L_{r,p}(i-1,j) + 1 \\ L_{r,p}(i,j-1) + 1 \\ L_{r,p}(i-1,j-1) + \delta_{r_i,p_j} \end{cases} \tag{3.15}$$

$$\delta_{r_i,p_j} = [r_i \neq p_j] \tag{3.16}$$

Трите случая в 3.15 отговарят съответно на делеции, инсерции и замествания.

Когато се използва разстояние на Хеминг след извършване на съпоставяне по двойки, се налага прилагане на процедури като алгоритъма на Нийдълман-Вюнш [104]. Там се разчита на сходството между две нуклеотидни бази или между две аминокиселини, чрез така наречената матрица на субституциите, която поставя различни „на-

казания“ за различните несъвпадения. Тук ще означаваме елементите ѝ със $S(r_i, p_j)$, където $r_i, p_j \in \mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}, \varnothing$. Разширението на S с размитата оценка за сходство отново не е много сложна операция.

Размито разстояние и размито съпоставяне по двойки

Предложената разширена функция на разстоянието ще използва оценка на увереност $s(r, i)$ като тези от 3.4.1 като размит индикатор на увереност. Нека за всяка секвенция r дефинираме размитото множество R_r, което означава сигурните позиции, и неговото допълнение E_r, което означава погрешните позиции, $E_r = cR_r$. За стандартното размито допълнение, тези множества биха били следните:

$$R_r(i) = s(r, i), \quad E_r(i) = cR_r(i) = 1 - s(r, i) \tag{3.17}$$

Всички функции на разстояние разчитат на двоична индикация на съвпадение и несъвпадение. При разстоянието на Хеминг H това е $[r_i \neq p_i]$, което е 1 при несъвпадение и 0 при съвпадение. По подобен начин, разстоянието на Левенщайн не само използва абсолютно същата индикация за съвпадения и несъвпадения, но също така измерва инсерциите и делециите с 1.

За да се използва апаратът на размитите множества за H, би следвало да се дефинира множество $X_{r,p}(i) = [r_i \neq p_i]$ и да използва операция, подобна на $R_p \cap R_r \cap X_{r,p}$ при дефиниране на разширените функции за разстояние. Тъй като обаче са в сила аксиомите за размити множества, а резултатът няма да зависи от избора на сечение, понеже оригиналните събираеми са 0 и 1, за простота ще използваме умножение и ще считаме, че размитите индикатори на увереност са тегла.

След като се модифицира функцията на Хеминг H от (3.13) да използва размитият индикатор, включващ множествата R_i, тя се трансформира в следното претеглено разстояние на Хеминг:

$$H^f(r, p) = \frac{\sum_i (R_r \cap R_p)(i)[r_i \neq p_i]}{\sum_i (R_r \cap R_p)(i)} \tag{3.18}$$

На пръв поглед не става ясно дали подобна модификация би свършила същата работа и за разстоянието на Левенщайн. След по-внимателен оглед на формулата за L се

забелязва, че третия случай от (3.15) е напълно идентичен на събираемите в разстоянието на Хеминг, и може да бъде модифициран аналогично. Другите два случая, които отговарят на инсерции и делеции, след хипотетично съпоставяне биха отговаряли на заместване с $\varnothing$ и могат да бъдат модифицирани по същия начин.

За да се избегне въвеждане на сложни размити множества, които се грижат за разминавания между двете секвенции поради липсата на съпоставяне, за дефиницията на претегленото разстояние на Левенщайн са използвани означенията за размита логика и конюнкция вместо сечение.

$$L^f_{r,p}(i,j) = \begin{cases} \sum_{k=1}^{i} R_r(k) & \text{if } j = 0 \\ \sum_{k=1}^{j} R_p(k) & \text{if } i = 0 \\ L^{fsbs}_{r,p}(i,j) & \text{otherwise} \end{cases} \tag{3.19}$$

$$L^{fsbs}_{r,p}(i,j) = \min \begin{cases} L_{r,p}(i-1,j) + R_r(i) \\ L_{r,p}(i,j-1) + R_p(j) \\ L_{r,p}(i-1,j-1) + \delta^f_{r_i,p_j} \end{cases} \tag{3.20}$$

$$\delta^f_{r_i,p_j} = R_r(i) \wedge R_p(j)[r_i \neq p_j] \tag{3.21}$$

За да се дефинира размито съпоставяне по двойки, трябва да се промени функцията на сходство $S(r_i, p_j)$. Всички промени дотук разчитаха на предположението, че несъвпадения, дошли от погрешни бази, ще бъдат пренебрегнати. За да се въведе същата зависимост за S, базите с ниска оценка за увереност трябва да се считат по-близки. С други думи това означава, че несигурно несъвпадение няма да наруши съпоставянето на базите на една и съща позиция.

При тези съображения, се въвежда следната функция:

$$S^f(r_i, p_j) = b\left(E_r(i) \vee E_p(j) \vee \tfrac{1}{b}\left(S(r_i, p_j) - a\right)\right) + a \tag{3.22}$$

Тук a и b са нормализиращи константи, които се използват, ако S приема стойности в интервала $[a, b]$ вместо в $[0, 1]$, което е практиката за матрицата на субституциите. Може да се провери, че това е еквивалентно на промяната в L^f и H^f. Нека означим събираемото в H с h и предположим, че матрицата на субституциите S означава не-

съвпаденията с 0, съвпаденията с b, а несигурните случаи със стойност, пропорционална на вероятността за съвпадение (например $\frac{1}{2}b$, $\frac{1}{3}b$ and $\frac{1}{4}b$). Тъй като законите на де Морган са в сила за използваната размита логика [52], следните твърдения са еквивалентни при използване на стандартното размито отрицание $\neg x = 1 - x$.

$$h(r_i, p_j) = 1 - \frac{1}{b}S(r_i, p_j) = \neg\frac{1}{b}S(r_i, p_j) \tag{3.23}$$

$$\begin{aligned} R_r(i) \wedge R_p(j) \wedge h(r_i, p_j) & \\ &= R_r(i) \wedge R_p(j) \wedge \neg\tfrac{1}{b}S(r_i, p_j) \\ &= \neg\left(\neg R_r(i) \vee \neg R_p(j) \vee \tfrac{1}{b}S(r_i, p_j)\right) \\ &= \neg\left(E_r(i) \vee E_p(j) \vee \tfrac{1}{b}S(r_i, p_j)\right) \\ &= \neg\left(E_r(i) \vee E_p(j) \vee \tfrac{1}{b}S(r_i, p_j)\right) \\ &= \neg\tfrac{1}{b}S^f(r_i, p_j) \end{aligned} \tag{3.24}$$

Макар че твърденията са еквивалентни, те не са непременно верни, понеже h приема само двоични стойности, докато S сама по себе си е размита оценка. За да са изпълнени равенствата, или в S трябва да има само две стойности (което е в сила за някои матрици за ДНК и РНК последователности, в които има само четирите бази и интервали), или трябва самото разстояние на Хеминг да се разшири да използва размитата оценка от S.

Въпреки това, ясно се вижда, че направената модификация в S е еквивалентна по своето действие. Когато в матрицата на субституциите има само стойностите 0 и b, претегленото разстояние на Хеминг H^f ще бъде равно на $\sum_{i=1}^{m}\left(1 - \frac{1}{b}S^f(r_i, p_j)\right)$.

3.6 Използване на невронни мрежи и други средства на изкуствения интелект

В основата си, задачата е за откриване на зависимост между данните и наличието на грешки. Тази зависимост разчита на много фактори, които са или неизвестни, или които не могат да бъдат лесно формализирани. В същото време, самите изследвани данни съдържат множество хаотични, неструктурирани и трудни за формализиране

зависимости. Апаратът на невронните мрежи е сериозен инструмент, който може да бъде използван за класифициране на базите на правилни и погрешни при тези условия, или да извежда числова оценка за вероятността те да бъдат правилно прочетени.

Това, което липсва за приложението на алгоритми използващи машинно самообучение, е същото като при валидацията – добро тестово множество. Един подход би бил да се приложат средствата използвани за изграждане на средствата за валидация, като мрежата се обучи с помощта на подхода за симулиране на грешки предложен в 3.4.2.2, но поради ниската честота на грешките по този начин не би могло да се конструира достатъчно голямо обучаващо множество. Поради това тук е използван алтернативен подход, който симулира голям брой виртуални грешки.

3.6.1 Вход

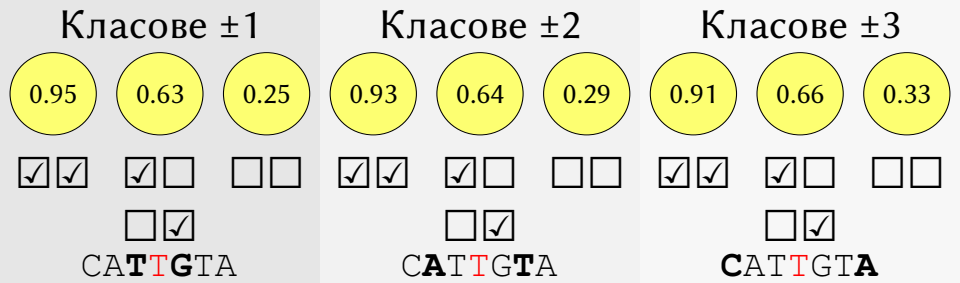
Дори при налични симулирани грешки, съставянето на входните данни остава значително предизвикателство, тъй като неопределеният брой секвенции, които се вземат при оценка на качеството на всяка разглеждана база, не се подава директно на обработка с невронна мрежа.

Разгледани са няколко подхода за обобщение на входните данни по критерии, които по подходящ начин ще представят съдържаната в прочитите информация, счетена за важна за откриване на грешките, по начин, запазващ значимите връзки.

3.6.1.1 Вход обобщен по сходство

Следвайки предположението, че два сходни прочита са по-важни за оценяване помежду си, отколкото два различни прочита, входните данни могат да бъдат обобщени по степента на сходство. Използвайки оценката за локално сходство $\mathrm{sim}(r, p, k)$ от (3.4) или от (3.5), за всяка оценявана база разделяме секвенциите в 10 групи, отговарящи на 10 различни нива на сходство. На всяка една от тези групи ще съответства един неврон, чийто вход ще е частта от секвенциите от групата, които потвърждават базата.

Такъв подход обаче не е особено подходящ. Самата мярка за сходство, която изглежда да има по-голямо значение за качеството на корекция на грешки, няма да подлежи на обучение. В същото време, резултатът ще е силно зависим от това дали сме



Фигура 3.6: Примерен вход за невронната мрежа

избрали (3.4) или (3.5) при разделяне на секвенциите. Освен това, така критерият на невронната мрежа може да стане прекалено близък до този на аналитичния подход, което може да навреди на възможостите двата метода да бъдат комбинирани. По тази причина е избран по-различен вариант за разделяне на входа.

3.6.1.2 Вход обобщен по сходство и отстояние

Критериите, по които сме формулирали нашите оценки за сходство в 3.4.1.2, са броят несъответствия в прозорец около оценяваната позиция и разстоянието до тези несъответствия. За да бъде тази информация частично видима за невронната мрежа, естествено би било вместо просто да се подава сходството, то да бъде разделено в групи по отстояние.

Така, вместо един класификатор по сходство, ще бъдат въведени няколко. Всеки класификатор съответства на дадено отстояние (± 1, ± 2, ...) от оценяваната позиция и разделя секвенциите на три групи – тези, които съвпадат напълно на съответното отстояние, тези, които съвпадат наполовина, и тези които напълно не съвпадат. Така на всяко отстояние отговарят с три входни неврона – тези на секвенциите, които съвпадат на 0%, 50% и 100% на съответните две позиции. Всеки входен неврон има за вход частта от секвенциите в групата от това разделение, които потвърждават оценяваната база. Пример за това е показан на фигура 3.6.

Първият неврон показва, че 95% от секвенциите, които съвпадат напълно с оценяваната на отстояние ± 1 от оценяваната позиция (всички секвенции, които имат `T` и `G` на съседни позиции), потвърждават базата T по средата на прозореца. Вторият неврон показва потвържденията от тези, които съвпадат само наполовина – имат само `T` или

само G, докато третият е за напълно несъвпадащите, които дават само 25% потвърждение на оценяваната база. Втората група неврони дават същата информация, но не за различията на отстояние 1, а за тези на отстояние 2, и съответното базите A и T, които заемат тези позиции.

Основен недостатък на подобно обобщение е, че невронната мрежа няма да „вижда“ напълно зависимостите между съседните позиции. От една страна, самият генетичен смисъл се съдържа вертикално в съответните последователности, от друга при определени експерименти беше установено, че вертикалната структура има значение за оценка на грешките – най-вече когато в прочитите се срещат т.нар. мононуклеотидни повтори, в които една и съща база се повтаря повече от веднъж. Това са грешки, към които секвениращите апарати са много чувствителни, и често увеличават или намаляват бройката на повторения.

Друг проблем е значението на взаимното срещане на несъвпадения. Острата оценка на сходство от (3.5) предполага, че две несъвпадения правят дадена секвенция много по-незначима за да бъдат взети предвид. Това би означавало, че потвърждения, идващи от секвенции, които имат едно несъвпадение, но на различни места, имат много по-голямо значение от секвенции, които имат по две несъвпадения. В определени ситуации двете могат да бъдат отразени по един и същ начин във входа, поради незапазване на вертикалните зависимости.

Една от основните насоки за разширение на тази невронна мрежа е премахването на посочения недостатък. Това би могло да стане с внасяне на допълнителни неврони, които дават информация за връзките между съседните позиции, или за промяна на архитектурата на невронната мрежа.

3.6.1.3 Рекурентна невронна мрежа и нестандартни архитектури

Рекурентните невронни мрежи позволяват подаване на вход с неограничена дължина. Тъй като те имат вътрешно състояние, би могло да се мисли за мрежа, която изследва всяка отделна потвърждаваща секвенция. Това обаче от една страна сериозно възпрепятства както обучението със стандартни алгоритми, а от друга силно се отличава от задачите, към които се прилагат такива невронни мрежи, при които входните данни представляват времева или друга последователност, в която реда има значение.

От друга страна ако се погледне (3.2), може да се види, че за всички двойки секвенции се прилага една и съща формула за сходство, и няма никакво основание тя да бъде различна. Ако всички сравнявани секвенции участват във входа, то всички трябва да бъдат оценени по един и същи начин. Това е една основа, която може да се използва за разработването на невронна мрежа, използваща или нестандартна топология, или разширение на рекурентните, и в същото време отговаря на условието няколко двойки секвенции да подлежат на една и съща трансформация.

Тъй като такива мрежи не се поддържат от използваната библиотека и тъй като представляват значително отклонение от теоретичната литература, тяхното изследване не бе сметнато за централна задача на този труд, но има достатъчен потенциал, за да бъде изследвано на един по-късен етап.

3.6.2 Обучение с виртуални грешки

Невронната мрежа трябва да може да класифицира правилно грешки, открити на всяка позиция в четенето, независимо от тяхното разпределение. Не е наложително да бъде взето предвид разпределението на грешките, поради което не е наложително да бъде използван подходът за симулация на грешки от 3.4.2.2. Разгледаният там подход се опитва да симулира неправилно прочетени бази по реалистичен начин, което означава, че с него обучаващото множество би било недостатъчно голямо.

За тази цел обучението бе извършено с множество, което симулира виртуални грешки на всяка възможна позиция. За всяка база в обучаващото множество се поставят по два примера – един с действителните входни статистики за базата според 3.6.1.2, класифициран като правилен, и втори със статистиките за случайна хипотетична база, която е заменила оригиналната, и е класифициран като погрешен. Това осигурява създаването на обучаващо множество с достатъчно по обем обучаващи примери, които са достатъчно представителни и съдържат грешки във вида, в който се очаква да бъдат срещнати действителните.

Тъй като броят на действителните грешки в данните е достатъчно малък, възможно е обучението да бъде извършено направо върху оригиналните данни след съпоставяне, без извършване на друга предварителна обработка. Понеже не се разчита на ин-

формация за действителната класификация на използваните данни, това наподобява методите за обучение без учител, но за разлика от тях класификацията на обуващите примери е част от симулацията, а не обект на предсказание, както е при методите за клъстерирането например. Въпреки това, този подход позволява при всяка процедура по премахване на грешките да бъде обучена нова невронна мрежа по множество от грешки, симулирани в изследваното множество данни.

3.6.3 Топология

Използвана е ациклична невронна мрежа с един скрит слой, като броя на неврони е избран емпирично, тръгвайки от половината на невроните във входния слой на мрежата. Използван е прозорец около оценяваната позиция с радиус от 10 бази, което означава, че мрежата има 30 входни неврона, като е направен опит да бъде обучена и мрежа по по-малки прозорци. Мрежата има един изходен неврон, който класифицира примерите като верни или погрешни.

3.6.4 Дървета на решенията и други подходи

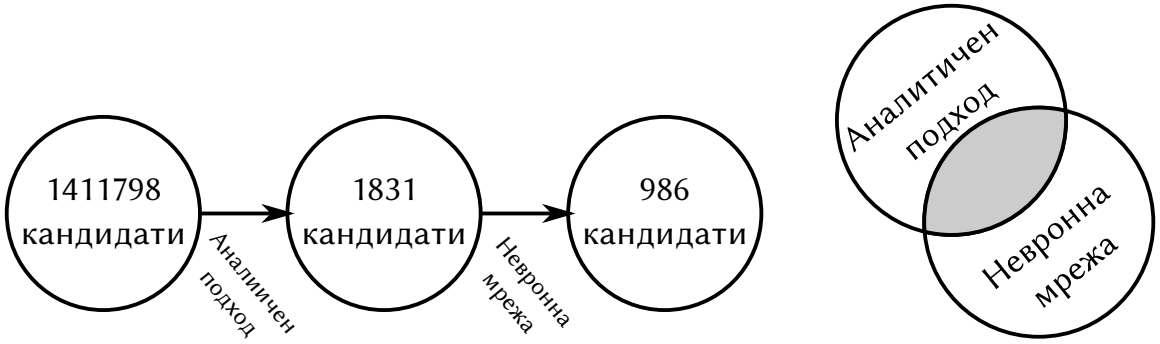
Могат да бъдат приложени и други подходи за класификация. При така избрания вид на входните данни, описан в 3.6.1.2, възниква въпроса дали е възможно да бъде създаден по-прост класификационен критерий, основан на серия от прагове върху честотите от различните входове. Поради това дърветата на решенията, например, могат да представляват обект на разглеждане.

Върху множеството за обучение бяха експериментално изпитани множество от методите, които са налични в системата за машинно самообучение Weka [156], като впоследствие моделът построен с помощта на случайна гора [20] бе приложен върху реалните данни и е включен за валидация в подхода, показан в 3.4.2.3.

3.6.5 Използване на получените модели

Един основен недостатък на обработваните данни при класификация на базите с методи на машинното самообучение е, че има голяма разлика между броя на правилните и неправилните бази. Ако правилните бази са милиони, то неправилните бази са

стотици, в най-добрия случай хиляди. Това означава, че дори моделът да има изключително добра точност, тя може да бъде недостатъчна, за да бъде приложен той върху всички данни. Например при милион бази за класифициране, дори 0.1% неточност би довел до хиляда некоректно идентифицирани грешки, което е повече от действителните грешки в данните.



(а) Верига на филтриране на потенциалните грешки (б) Използвано сечение

Фигура 3.7: Комбинация на аналитичния подход с невронна мрежа

По тази причина моделите са използвани като допълнителен класификатор. Те са приложени чак след като е бил използван някакъв друг критерий за избор на потенциални грешки, които да бъдат подложени на по-прецизна класификация. В случая първоначалният критерий е аналитичният подход, разгледан в 3.4.1. Получената комбинация е илюстрирана на фигура 3.7, където се вижда как аналитичния подход подбира ограничен набор от потенциални грешки сред множеството на всички бази в прочитите, след което невронната мрежа бива използвана за допълнителното му ограничаване (до сечението между множествата грешки предсказани от двата подхода).

Тъй като и аналитичният подход, и методите на машинно самообучение разчитат на сходството между секвенциите и на честотите на срещане, може да има голяма степен на съвпадение между откритите грешки, което допълнително да затрудни приложението на тези методи дори в посочената ситуация. Откриването на независими критерии е една възможност за бъдещо развитие на разгледаните тук предложения.

Глава 4

Софтуерна реализация

Както предварителната обработка, така и извършеният експериментален анализ за оценка на предложените в този дисертационен труд методи, наложиха разработването на множество от софтуерни инструменти за тяхната автоматизация. Беше направен извод, че обединяването на тези инструменти в обобщена система за изпълнение на работни потоци значително би улеснило бъдещи изследвания. За целите на това обединение беше създаден език за описание на работни потоци на основата на езика YAML, който е реализиран като част от софтуерен пакет, наречен untangle.

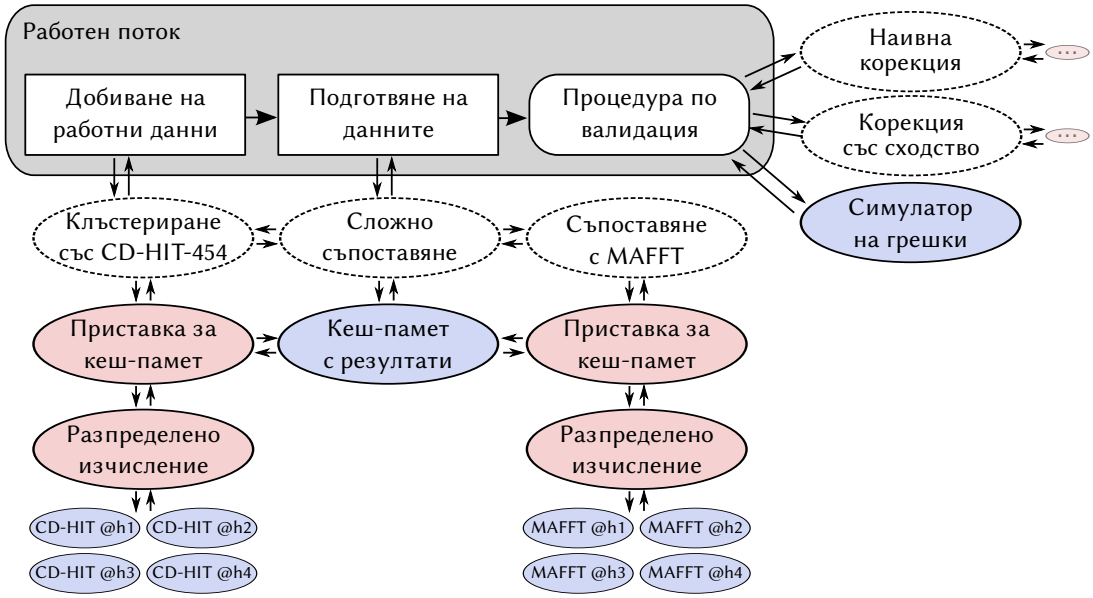
Тук е представено описание на създадения език, както и информация за софтуерната му реализация.

4.1 Език за описание на управляеми работни потоци

След прочитане на данните с апарата за секвениране, те подлежат на редица обработки, преди те да бъдат подложени на същинско научно изследване. Това включва стандартни операции като клъстериране [79], съпоставяне [145] и изчистване на грешките [144]. По време на тяхното изпълнение се комбинират различни софтуерни инструменти, които трябва да се настроят за изпълняваната задача, а при някои експерименти (особено в метагеномиката), се налага да бъдат комбинирани по специфичен начин.

За целите на този труд е разработвана софтуерна система за изпълнение на управляеми работни потоци, която цели да следва модела за изпълнение показан на фигу-

ра 4.1. Тя включва разработените методи за откриване на грешки предложени в 3.4.1, методите за изграждане на матрица на разстоянията с размитият индикатор на увереност предложени в 3.5, и поддържа работа с външни инструменти за клъстериране и съпоставяне. Тя има за цел да е гъвкава и широко приложима, така че да може да се използва за всякакви експерименти в областта на метагеномиката, а на по-късен етап и за други биоинформатични задачи.



Фигура 4.1: Моделно изпълнение на работен поток

Ключов елемент от разработваната софтуерна система са средствата за описание на работните потоци и средствата за описание на самите операции в тях. Вторите са обособени в система, която има сходства с информатични средства като прототипите и интерфейсите, които се срещат в някои езици за програмиране, но е насочена към създаване на помощни средства за откриване на съвместимите операции и включването им в работни потоци, както и за извършване на необходими преобразувания на данните по време на изпълнение.

Описанията на типовете данни и на операциите са средства, чиято помощ може да бъде използвана при проектирането на работни потоци. Те могат да се използват както в инструменти за тяхното конструиране, така и в инструменти за тяхното изпълнение и валидиране, а освен това ще изпълняват и ролята на източник на техническа документация. С достатъчно гъвкави средства за описанието на самите работни по-

тоци, в които самите потоци или части от тях могат да се разглеждат като операции, тази информация може по автоматичен начин да се пренесе и в тяхното описание.

4.1.1 Описание на типовете данни

Описанията на типовете данни и преобразуванията между тях са изградени на основата на системата класове в езика Python [26]. В Python всеки клас A може да има произволен брой бащини класове и произволен брой преки наследници. Клас B се счита подклас (subclass, наследник) на A, ако B съвпада с A, или B има за бащин клас подклас на A.

За целите на създаване и валидиране на биоинформатични работни потоци, както и за целите на автоматично преобразуване на данните, е използвана колекция от специални абстрактни класове[1] (*), и голям набор от техни разширения, с които са описани използваните типове данни, различни техни комбинации и някои зависимости между тях. За целта са използвани Python Abstract Base Classes (ABC) [126, 161].

Формално е обърнато внимание основно на разширенията, които са внесени. Това е така, защото тук от класовете на Python е използвана основно възможността за изграждане на йерархия, без да са включени средства, които те предлагат за разработване и разширяване на софтуерни инструменти. Поради това, дефиницията на един абстрактен клас в Python може да се сведе до списъка от неговите бащини класове, което формално може да се запише по следния начин.

$$A : A \subset B_1, A \subset B_2, \ldots, A \subset B_n \tag{4.1}$$

Такова множество от класове $\{A_i\}$ могат да участват в две релации – релация *подклас*, обозначена тук с $A_i \subset A_j$, и релация *принадлежност* към клас, обозначена тук с $x \in A_i$.

[1] Значението на абстрактен клас в Python е различно от това в другите езици за програмиране. Позволена е имплементация на методи (в противоречие на очакването за абстрактен клас), изрична регистрация на подкласове, които не са програмно свързани (по подобие на интерфейсите в други езици), и дори използването на произволни изрази за дефиниране на класовата йерархия. Дефинираните тук класове обаче са напълно абстрактни, тъй като нямат реална функционалност.

Ако изключим транзитивното им свойство, тези релации *обикновено* се дефинират изрично – всеки клас се дефинира със списък на своите бащини класове и всеки екземпляр (instance) на клас се дефинира с класа, към който той принадлежи. Гъвкавата Abstract Base Classes (ABC) система [126] на Python позволява тези релации да бъдат разширени и предефинирани, така че даден обект може да е от някой клас, без непременно да е негов действителен екземпляр, а даден клас може да е наследник на друг, без непременно да го включва в бащините си класове. Възможно е тези релации да бъдат предефинирани с произволни изрази.

Използвайки този гъвкав апарат са разработени няколко разширения, които внасят нови скрити правила между специалните категории от класове (*) (от страница 95) използвани за описание на връзките между типовете данни.

Дефиниция 4 (Класов поделемент). Въвежда се клас $A[B]$, обозначаващ подкласа на класа от контейнери A, чиито елементи съдържат само поделементи от класа B. Елемента x принадлежи на клас $A[B]$ тогава и само тогава, когато x принадлежи на A и поделементите на A принадлежат на B. Съответно, $A'[B']$ е подклас на $A[B]$ тогава и само тогава, когато A' е подклас на A и B' е подклас на B. Така $A[B]$ е дефиниран като клас, чиито релации принадлежност и подклас изпълняват следните условия:

$$x \in A[B] \Leftrightarrow x \in A \wedge \forall i(x_i \in B) \tag{4.2}$$

$$A'[B'] \subset A[B] \Leftrightarrow A' \subset A \wedge B' \subset B \tag{4.3}$$

Класът $A[B]$ е абстрактен, няма действителни екземпляри или функционалност отвъд тези релации. Класът $A[B_1, B_2, B_3]$ е дефиниран по подобен начин като класовете контейнери A, които съдържат тройки елементи от класовете B_1 B_2, B_3.

$$x \in A[B_1, B_2, \ldots, B_n] \Leftrightarrow x \in A \wedge \forall i \forall j(x_{i,j} \in B_j) \tag{4.4}$$

$$A'[B'_1, B'_2, \ldots, B'_n] \subset A[B_1, B_2, \ldots, B_n] \Leftrightarrow A' \subset A \wedge B'_1 \subset B_1 \wedge \ldots \wedge B'_n \subset B_n \tag{4.5}$$

Класовият поделемент е асоциативна двоична операция между специалните класове (*) въведени на страница 95. Това означава, че $A[B[C]]$ е еквивалентно на $A[B][C]$.

Това е следствие на интуитивното очакване, че A контейнер съдържащи B контейнери съдържащи C елементи е същото като $A[B]$ контейнери съдържащи C елементи.

Дефиниция 5 (Операции между класове, взаимствани от множества). Въвеждат се класовете $A \cup B$, $A \cap B$, $\neg A$, които обозначават съответно обединение, сечение и допълнение на класове. X е подклас на $A \cup B$, ако X е подклас на един от класовете A или B; подклас на $A \cap B$, ако е подклас едновременно на A и B; или подклас на $\neg A$, ако не е подклас на A. Принадлежността се дефинира по аналогичен начин.

За да е пълна системата и да са дефинирани всички операции, трябва да се добавят следните изрични дефиниции между класовете получени чрез операциите за множества.

- $\neg A$ е подклас на $\neg B$, ако B не е подклас на $\neg A$.
- $A \cup B$ е подклас на $C \cup D$, ако както A, така и B са подкласове на $C \cup D$.
- $A \cap B$ е подклас на $C \cap D$, ако или A, или B е подклас на $C \cap D$.

В стандартната реализация на предложената система, операциите са дефинирани с достъпните символи за оператори – ‘`A | B`’, ‘`A & B`’ и ‘`~A`’, които в Python стандартно се използват за обединение, сечение и допълнение[2] на множества.

Дефиниция 6 (Константни класове). Константните класове са класове, които съдържат само един единствен константен елемент като свой екземпляр, например $C = \{4\}$.

Класовете дефинирани с операция поделемент, операции от множества, както и константните класове, се считат за съставни класове. В софтуерната реализация всеки несъставен клас, регистриран в типовата система от специални класове (*) (от страница 95), позволява набор от произволни Python класове да бъдат регистрирани като негови наследници. Могат да бъдат регистрирани и самите абстрактни класове, били те несъставни или съставни, а за съвместимост, и интерфейси на софтуерната библиотека Zope.[3]

[2] Вградените множества не поддържат допълнение, но използването на ‘`~A`’ е стандартна практика в много външни библиотеки, и се използва от т.нар. побитови множества от флагове, които се дефинират на основата на побитовите оператори.

[3] Zope интерфейси се използват значително от софтуерната рамка Twisted, върху която е изграден разработеният софтуер.

Например, в системата е дефиниран класът $Sequence$ и вградените Python типове $list$, $tuple$ са регистрирани като последователности в него, което известява типовата система, че всеки списък (list) или n-орка (tuple) също е и $Sequence$ по същия начин, по който това е направено в Python ABC [126].

Най-често използваният тип данни в представената система ще бъде $BioSequences$, който сочи към колекция от нуклеотидни или аминокиселинни последователности, които са съхранени на диска в произволен формат, на някакъв мрежов ресурс или в оперативната памет. Естествено съвместим тип би бил Python списък, който съдържа отделни такива секвенции. Така би било разумно да се окаже, че $list[BioSequence]$ и $BioSequences$ са подобни и съвместими, и конвертирането между тях да се извърши автоматично.

4.1.2 Услуги, прототипи на методи и функции

С наличната специална система от класове (*) (от страница 95) става възможна декларацията на сигнатури на функции и методи, които изпълняват биоинформатични операции и участват в работните потоци. Методите представляват функции, които са организирани в дърво от услуги и техни изпълнители.

Начинът на *вътрешно* описание на сигнатурите на методи може да се различава между имплементациите. Тук са дефинирани единствено средствата за деклариране на нови *външни* услуги, които използват езика YAML и могат да бъдат споделени между различни софтуерни инструменти. Строго са дефинирани единствено имената на услуги и методи, които се използват във всички дефинирани в този труд формати за обмен на описани работни потоци и операции. В тези описания името на услугата и на метода са разделени с две точки, и могат да съдържат следните позволени символи.

⟨*service*⟩ ::= ⟨*letter*⟩ ⟨*letter*⟩ | ⟨*digit*⟩ | - ; ⟨*method*⟩ ::= (⟨*letter*⟩ | _) ⟨*letter*⟩ | ⟨*digit*⟩ | _ ;

Сигнатурите по-долу са представени в Python-подобен псевдокод. Изразите, декларациращи типовете използват синтаксиса, предложен в 4.1.1, докато типовете на аргументите и връщаните стойности се отделят от аргументите и методите с две точки, както в Python 3. За функции, използваният запис на псевдокод е идентичен със син-

таксиса на Python 3[4], и се отличава единствено в позоваването на име на услуга.

4.1.2.1 Прототип на функция

Прототипът на функция задава сигнатура на група изпълними обекти, които са записани под определено име, получават подредени именувани аргументи от определени класове данни като свой вход, и връщат определен клас данни като свой изход.

Аргументите са именовани и подредени, които означава, че в кода на работните потоци те могат да бъдат позовани както по име, така и по ред. Конвенцията е позоваване по ред да се използва за методите от услугата ‘`core`’, описана в 4.1.5, както и за прилежащите помощни методи, които приемат един или два аргумента със строго дефинирана наредба (като ‘`count`’). За всички останали случаи аргументите трябва да се позовават по име. Това подобрява яснотата при функции с голям брой аргументи и позволява позоваването на конкретен аргумент при функции с незадължителни аргументи.

Тук е показан един примерен прототип на функция.

```
align(sequences: BioSequencesLike): BioSequences
```

В Python, методите са обикновени функции, които са прикрепени към клас или негов екземпляр, така че естествения начин да се дефинира прототип на метод е прототипът на функция да се прикрепи към екземпляр на клас. В разработваната система не се работи с класовете и техните екземпляри директно. Вместо това се създават услуги (services), които представляват разширени интерфейси, включващи произволен брой екземпляри, наречени изпълнители (providers). Прототипите на методите са прикрепени към услугите, които са осигурени от множеството свои изпълнители.

4.1.2.2 Прототип на метод

Прототипът на метод е прототип на функция, който се намира в тялото на класа дефиниращ дадена услуга.

Показан е примерен такъв прототип.

[4] С изключение на това, че международни символи не са позволени в идентификаторите, за да може да бъде използван Python 2, както и други езици, които не поддържат международни символи в синтаксиса си

```
alignment—multiple:align(sequences:
                         BioSequences | Sequence[BioSequence]): BioSequences
```

4.1.2.3 Услуга

Услугата е именувана колекция от прототипи на методи. Всеки изпълнител на услугата трябва да предлага всички методи, които имат прототипи в услугата. Тези методи трябва да съвместими с прототипите, като приемат поне аргументите, които са описани в тях, и има съвместимост на типовете на входните и изходните данни.

Всички изпълнители се регистрират към услугата, която позволява извличането на произволен свободен изпълнител.

Регистрацията на изпълнители е това, което отличава услугите от интерфейсите в езици като Java. Всъщност, от гледна точка на софтуерната реализация, прототипите имат второстепенна важност и се използват при проектирането на работните потоци, както и като документация. Задача на самия изпълнител е да гарантира, че изпълнява необходимите прототипи. Автоматичния избор на изпълнители със стандартен ППИ (API) е тяхната основна софтуерна функция. Самите изпълнители остават скрити зад този ППИ. Услугите представляват документиран, строго деклариран инструмент за до достъп до различни изпълнители, които ги предлагат.

4.1.3 Управляеми работни потоци

Описанието на работния поток става чрез деклариране на колекция от целеви множества от данни (цели). Множествата са взаимозависими и могат да бъдат или явни, насочени към ресурс като файл на диска, или неявни, получени като резултат от прилагане на операция (метод от услуга или друг работен поток) върху някое от другите множества. Освен обикновени операции, може да се изпълняват и някои функции от по-висок порядък като *map*[5] и *reduce*.[6]

Работните потоци се описват с помощта на езика за обектна нотация YAML [13]. При зареждане работния поток с него се асоциира услуга и по подобие на методи-

[5] Прилага друга функция върху всички елементи от последователност

[6] Прилага друга функция върху двойки стойности, вземани от една последователност докато тя бъде сведена до един елемент (като $\sum$ и $\prod$)

те всяко декларирано множество се сдобива с прототип, който описва всички входни данни необходими за да бъде получено, и самото множество като изходни данни. Това позволява правенето на връзки между няколко работни потока.

Тук са описани накратко следните категории множества от данни, които са поддържани като час от работния поток.

4.1.3.1 Входове

Входните стойности или множества от данни трябва да са достъпни преди изпълнението на работния поток. Примери за такива са файл, Интернет адрес, като резултат от изпълнението на друг работен поток, и вход от потребителя. Те образуват параметрите (аргументи и опции) на работния поток.

```
source: {category: input, optional: false, type: biosequences, key: source}
```

Дадения примерен YAML код декларира вход именуван ‘`source`’, чийто съдържание ще бъде достъпно вътре в работния поток под това име. По подразбиране, същото име се използва и за параметъра, който подава съдържанието, но то може да бъде променено с помощта на дефиниционния ключ ‘`key`’.

Стойност по подразбиране може да бъде указана чрез ключа ‘`default`’ когато ‘`optional`’ има стойност ‘`true`’. Сложни множества от данни също може да бъдат незадължителни, но тяхното съдържание по подразбиране не може да бъде указано вътре във файла – допустими са само стойностите ‘`null`’ (по подразбиране) или чрез отпратка към друго множество чрез специалния YAML тип ‘`!ref`’ (описан в 4.1.4).

Така дефинираният вход има тип ‘`BioSequences`’.[7]

4.1.3.2 Цели, генерирани чрез проста операция

Изчислените и генерираните стойности и множества от данни не съществуват при започване на изпълнението на работния поток, а се генерират от останалите с помощта на изпълнение на една операция. По подобие на системата Make [4], те биха могли да бъдат съхранени в кеш, но за разлика от нея тук не съществува начин те да бъдат подадени от потребителя.

[7] Типовете във софтуера и йерархията използват CamelCase, но за по-добра четимост в описателните файлове на работните потоци и операциите са изписани с малки букви.

Те се декларират чрез пълното име на генериращия ги метод и неговите аргументи.

```
aligned:
    category: operation
    method: alignment—multiple:align
    arguments: {sequences: !ref source, quality: 0}

clustered:
    category: operation
    method: clustering:cluster
    arguments: {sequences: !ref aligned,
                threshold: !optional—input threshold}
```

В тези два примера се декларират две множества от данни – целта ‘`aligned`’ се генерира с метода ‘`align`’ на услугата ‘`alignment-multiple`’, който избира подходящ софтуер за множествено съпоставяне,[8] който е достъпен на изпълняващия компютър, или при използване на реализация с разпределено изпълнение – на друг компютър, който не е натоварен. Метода ‘`align`’ ще получи множеството данни ‘`source`’ от 4.1.3.1 като аргумент ‘`sequences`’ с помощта на YAML типа ‘`!ref`’, използван за отпратки и разгледан по-подробно в 4.1.4. Аналогично, резултата от изпълнението му ще се подаде като аргумент на метода ‘`cluster`’.

4.1.3.3 Цели, генерирани чрез map операция

Специалната категория *map-operation* позволява да се декларират множества, които се получават от последователност от стойности (или множества) след многократно прилагане на метод от някоя услуга, резултата от което е последователност от резултатите получени от съответната операция. Входните данни трябва да са от някой от следните типове: $Sequence$, $ItemStore$, или $ItemGenerator$.

```
realigned:
    category: map—operation
    iterate—over: !ref clustered
    method: alignment—multiple:align
    arguments: {sequences: !current}
```

[8] В действителен работен поток би бил позован конкретен софтуер за множествено съпоставяне, защото избора на такъв е съществен за всяка едно изследване. Това е обсъдено в повече подробности в 4.1.2.

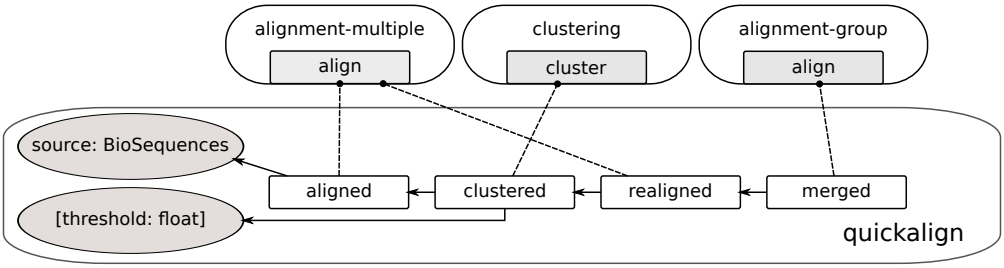
Специалния тип ‘`!current`’ оказва текущия елемент в итерацията при прилагане на операцията.

4.1.3.4 Цели, генерирани чрез reduce операция

По подобие на *map-operation*, специалната категория *reduce-operation* позволява да се декларира неявно множество, което се получава от последователност от стойности (или множества) след многократно приложение на двуаргументен метод от някоя услуга. Методът се прилага върху предишния резултат, получен от приложение на метода, и следващата стойност от последователността, а крайният резултат е последната получена стойност. (Например, функцията сума може да се представи като reduce операция от функцията прибавяне и последователността, която се сумира – ‘`sum(L) = reduce(add, L)`’.) Входните данни трябва да са от някой от следните типове: $Sequence$, $ItemStore$, или $ItemGenerator$.

Следния пример използва групово съпоставяне, за да слее всички съпоставени клъстъри от ‘`realigned`’ в голямо множество, което съдържа всички входни съпоставени в едно голямо множество.

```
merged:
    category: reduce-operation
    iterate-over: !ref realigned
    method: alignment-group:align
    arguments: {sequences1: !left, sequences2: !right}
```



Фигура 4.2: Примерен работен поток “quickalign”

Ако всички дадени примери се слеят в един работен поток, наречен ‘`quickalign`’, както е показано на фигура 4.2, то множеството ‘`merged`’ е достъпно като метод на услуга (виж 4.1.6.2) със следния прототип:

```
quickalign:merged(input: BioSequencesLike): BioSequences
```

4.1.4 Отпратки към стойности и множества данни

Както е показано на примерите по-горе, стойностите и множествата данни от работния поток могат да бъдат назовани чрез специалния YAML тип ‘`!ref`’. За да бъде улеснена реализацията на софтуерни решения за изпълнение и инспекция на предложения формат, такива отпратки са позволени единствено като аргументи на методи и, по изключение, като стойности по подразбиране за входовете.

4.1.5 Език за изрази

4.1.5.1 Синтаксис на изразите

За описание на сложни условни зависимости и разклонения в работните потоци е въведен език за изрази. Той е описан с помощта на безконтекстна граматика, която е компилирана с LR(1) *синтактичен анализатор* (parser). Граматиката е изградена по подобие на изразите в C, използвайки стандартните C оператори, допълнени с оператор за степенуване и с разширена азбука за идентификатори. Използвайки разширена[9] форма на Бекус-Наур, граматиката може да бъде описана по следния начин.

⟨*integer*⟩ ::= ⟨*digit*⟩ { ⟨*digit*⟩ } ; ⟨*float*⟩ ::= ⟨*digit*⟩ { ⟨*digit*⟩ } . { ⟨*digit*⟩ } ;

⟨*identifier*⟩ ::= (⟨*letter*⟩ | _) { ⟨*letter*⟩ | ⟨*digit*⟩ | _ | : | - } ;

⟨*dataset*⟩ ::= ⟨*identifier*⟩ ; ⟨*function call*⟩ ::= ⟨*identifier*⟩ ([⟨*expr*⟩ { , ⟨*expr*⟩ }]) ;

⟨*unary*⟩ ::= (~ | - | +) ⟨*expr*⟩ ;

⟨*multiplicative*⟩ ::= ⟨*expr*⟩ (* | / | // | %) ⟨*expr*⟩ ; ⟨*additive*⟩ ::= ⟨*expr*⟩ (+ | -) ⟨*expr*⟩ ; ⟨*power*⟩ = ⟨*expr*⟩ ** ⟨*expr*⟩ ;

⟨*bitwise and*⟩ ::= ⟨*expr*⟩ & ⟨*expr*⟩ ; ⟨*bitwise xor*⟩ ::= ⟨*expr*⟩ ^ ⟨*expr*⟩ ; ⟨*bitwise or*⟩ ::= ⟨*expr*⟩ | ⟨*expr*⟩ ;

⟨*shifts*⟩ ::= ⟨*expr*⟩ (>> | <<) ⟨*expr*⟩ ; ⟨*comparisons*⟩ ::= ⟨*expr*⟩ (< | <= | > | >= | == || !=) ⟨*expr*⟩ ;

[9] С квадратни скоби са означени незадължителните елементи, с фигурни скоби са означени елементи, които се повтарят нула или повече пъти, кръглите скобите се използват за групиране, а правилата се терминирани с точка и запетая.

⟨*logical not*⟩ ::= `!` ⟨*expr*⟩ ; ⟨*logical and*⟩ ::= ⟨*expr*⟩ `&&` ⟨*expr*⟩ ; ⟨*logical or*⟩ ::= ⟨*expr*⟩ `||` ⟨*expr*⟩ ;
⟨*conditional*⟩ ::= ⟨*expr*⟩ `?` ⟨*expr*⟩ `:` ⟨*expr*⟩ ;

⟨*expr*⟩ ::= ⟨*atom*⟩ | ⟨*power*⟩ | ⟨*unary*⟩ | ⟨*multiplicative*⟩ | ⟨*additive*⟩ | ⟨*shifts*⟩ | ⟨*bitwise and*⟩ | ⟨*bitwise xor*⟩
| ⟨*bitwise or*⟩ | ⟨*comparisons*⟩ | ⟨*logical not*⟩ | ⟨*logical and*⟩ | ⟨*logical or*⟩ | ⟨*conditional*⟩ ;
⟨*atom*⟩ ::= ⟨*dataset*⟩ | ⟨*integer*⟩ | ⟨*float*⟩ | ⟨*group*⟩ | ⟨*function call*⟩ ; ⟨*group*⟩ ::= `(` ⟨*expr*⟩ `)` ;

Трябва да се отбележи, че тази формална граматика не дефинира приоритета на операторите, който би могъл да се опише с Бекус-Наур форма следвайки алгоритъма използван от Мартин Ричардс и Колин Уитби-Стивънс в [119] за компилатора на програми на езика BCPL и за обработка на приоритета в изходния им код. Такъв запис би усложнил разбирането на синтаксиса, а в същото време може да се замени с използване на граматика без приоритети и последващо пренареждане на полученото дърво в съответствие с тези приоритети, което представлява и по-ефективен алгоритъм за анализ [45]. На друг още по-ефективен алгоритъм разчитат и LR синтактичните анализатори [69], какъвто е използван тук.

Поради това приоритета и посоката на групиране на асоциативните оператори са описани отделно със средствата на използвания LR(1) синтактичен анализатор – lrparsing за Python[10]. Групите синтактични елемени от ⟨*expr*⟩ имат различни приоритети, намаляващи според подредбата им в описанието на граматиката. Във всички други случаи синтактичните елементи имат еднакъв приоритет. Всички оператори са ляво-асоциативни, с изключение на степенуването, унарните оператори и побитовите оператори, които са дясно-асоциативни.

4.1.5.2 Използване на изрази

Вътре в описанията на работните потоци изразите могат да се използват като аргументи на операции на мястото на отпратки към стойности и множествата от данни. За тази цел е деклариран специалния YAML тип '`!expr`'. При зареждането на работния поток, всеки израз в него бива компилиран. В компилиран вид изразите представляват множество от цели, които се добавят към работния поток и стават част от йерархията от множества от данни. Изчислението на израза става по същия начин, по който

[10] `https://pypi.python.org/pypi/lrparsing/1.0.7`

и изпълнението на съответстващия му работен поток – изрази и работни потоци са взаимозаменяеми.

Даден е пример за израз, който използва условния оператор. Условието в него е проверка, дали множеството от данни има повече от 3000 елемента. В резултат на изпълнение на израза ще се генерира само целта ‘processed_quick’, ако условието е изпълнено, или само целта целта ‘processed_quality’ в противен случай.

```
counting:count(dataset) >= 3000 ? processed_quick : processed_quality
```

При компилиране на този израз се получава следното множество от цели в работния поток, които биват добавени в йерархията му, а самия израз се превръща в отпратка към последния.

```
_count_dataset_x002: {
  arguments: [!ref dataset],
  category: operation
  method: counting:count

_count_data_ge_3000_x001:
  arguments:
    [!ref _count_dataset_x002, 3000]
  category: operation
  method: core:ge

_if_count_data_processed__processed__x000: {
  arguments: [!ref '_count_data_ge_3000_x001',
              !ref 'processed_quick', !ref 'processed_quality'],
  category: operation, method: core:condition }
```

Втория и третия аргумент на ‘core:condition’ се изчисляват при поискване от самия метод според стойността на условието, поради което след изпълнение на този работен поток само един от тях бива бъде генериран.

4.1.6 Деклариране на външни услуги

Взаимозаменяемостта между работни потоци и услуги от 4.1.6.2 дава един начин за деклариране на нови услуги, които дават достъп до комбинация от методи на други достъпни услуги. За осигуряване на много по-големи възможности за разширение е необходима възможност за деклариране на услуги, използващи произволни външни инструменти. За тази цел се използват шаблони за деклариране на нови услуги и техни изпълнители. Всеки шаблон е свързан с дадена услуга, към която се регистрират

всички операции, които го използват.

Следният пример показва декларация на две услуги за съпоставяне, които използват външната програма MAFFT [67].

```
alignment_multiple_mafft:
  factory_name: alignment_multiple_cmdline
  output_on_stdout: true
  commandline: mafft {options} {input}
  options:
    nofft: !switch --nofft,
    ep: !option:int --ep {0:d}

alignment_group_mafft:
  factory_name: alignment_group_cmdline
  output_on_stdout: true
  commandline: mafft_profile {left} {right}
```

Първият пример използва шаблона ‘alignment-multiple-cmdline’, който създава нова подуслуга на ‘alignment-multiple’, която ще бъде именована ‘alignment-multiple-mafft’. При извикване на метода ‘align’ се създава временен файл, в който се записват входната последователност от ДНК секвенции, след което се извиква изпълнимият файл ‘mafft’ с този файл като аргумент, след което съпоставените секвенции се изчитат от стандартния изход на MAFFT.

Ключът ‘options’ декларира незадължителни параметри на метода, които се превеждат в съответните незадължителни параметри на изпълнимия файл. В случая това са булевият аргумент ‘nofft’, който при стойност истина добавя опцията ‘--nofft’ към командния ред, както и целочисленият аргумент ‘ep’, който се подава форматиран чрез опцията ‘--ep’ на командния ред.

По подобен начин втория пример декларира услуга за групово съпоставяне на две множества от секвенции – ‘left’ и ‘right’, които вече са били съпоставени поотделно.

4.1.6.1 Деклариране на нови шаблони

Услуги и изпълнители могат да се декларират и чрез допълнително създадени шаблони, за да бъдат описани напълно произволни услуги, тъй като системата позволява декларация на произволни шаблони. Следният пример би декларирал използвания по-горе шаблон за групово съпоставяне, ако той не беше вграден в системата.

```
alignment_group_cmdline:
    service_name: alignment_group
    operation_method_name: align
    inputs: {left: left, right: right}
    input_types: {left: biosequenceslike, right: biosequenceslike}
    input_convert: {left: biosequences, right: biosequences}
    output_types: {output: biosequences}
    return_output_name: output
    default_input_formats: {left: fasta, right: fasta}
```

Това декларира нов шаблон ‘`alignment-group-cmdline`’, който съответства на услуга ‘`alignment-group`’ с единствен метод ‘`align`’ с аргументи ‘`left`’ и ‘`right`’. Когато шаблонът бъде използван и към него бъде приложен непопълнен образец на команден ред, той ще бъде попълнен според заместванията, дефинирани в ключа ‘`inputs`’. Ако той отсъства, той ще бъде автоматично попълнен от останалите ключове. За шаблона са описани типовете на аргументите и изходите, както и желаният краен тип и файлов формат, който ще бъде записан във файл или прочетен от файл. При използване на шаблона, форматът на входовете и изходите може да бъде променен, но не и типът.

4.1.6.2 Работен поток и услуги

Работните потоци и услугите са взаимозаменяеми. Всеки работен поток, който се зарежда в системата, автоматично се регистрира като услуга и като неин изпълнител. Всяко целево множество от работния поток може да бъде генерирано чрез извикване на метод от услугата, който има същото име и приема всички входове на работния поток като свои аргументи.[11] Едно от целевите множества би следвало да бъде основно, но не са наложени изкуствени ограничения, които да гарантират това.

Тази възможност позволява не само да бъдат декларирани сложни услуги, които са представени от цял работен поток. Тя дава свободата да бъдат създавани рекурсивни работни потоци, които могат да извикват себе си по време на изпълнението си, както и системи от работни потоци, които се позовават един друг, включително и в неявна рекурсия.

[11] Това е един от случаите, в които наредената позоваване на аргументи не бива да бъде използвано. Стандартната реализация използва *нестандартно* разширение на YAML, което запазва подредбата на входовете, но в YAML библиотеките на други езици за програмиране това не е възможно и използването на реда би нарушило съвместимостта.

Примерът, показан на фигура 4.2 и описан в 4.1.3.2 съпоставя множество секвенции като го клъстерира, съпоставя отделните клъстери и после ги слива с групово съпоставяне. Това би могло да бъде разширено с помощта на рекурсия, която разбива множеството на клъстери, докато се стигне до клъстер, който е достатъчно малък, за да бъде съпоставен. Това би позволило съпоставянето на масивни множества от данни, за сметка на качеството на получения резултат (разделянето на множествата влошава качеството на съпоставяне – намирането на най-доброто съпоставяне е NP-трудна задача, което налага неточни евристични методи, а допълнителното клъстериране допълнително допринася за тази неточност).

4.2 Софтуерна система untangle

Представената система за изпълнение на работни потоци е реализирана като част от софтуерния пакет untangle, използващ Python версия 2.7. Използвана е асинхронната софтуерна рамка Twisted [163], която позволява както паралелно изпълнение на различни инструменти за разработка, така и бъдещо развитие, включващо разпределено изпълнение, отдалечено управление и интеграция с мрежови услуги. Включени са инструменти за команден ред, които изпълняват описаните работните потоци.

4.2.1 Модули

Софтуерната реализация предлага модулна структура, която позволява бъдещо разширение, използване на инструментите включени в нея от външен софтуер, както и модел за изграждане на алтернативни реализации на предложените в този труд методи и решения.

Регистър от услуги `untangle.service`. Регистърът от услуги поддържа колекцията от услуги и техни методи, които са достъпни за използване от работните потоци. Регистърът е достъпен от пакета ‘`untangle.service`’, и може да бъде използван направо, без регистрация на изпълнители, когато задачата не изисква изпълнение – например при реализация на софтуер за визуализация.

Пакет с вградени изпълнители `untangle.provider`. Пакетът ‘`untangle.provider`’ съдържа стандартния интерфейс за дефиниране на нови изпълнители, който включва основните средства необходими за откриване на най-подходящия изпълнител, както и за неговото изпълнение – като семафори за управление на паралелното изпълнение.

Четене и зареждане на външни операции `untangle.operation`. Пакетът ‘`untangle.operation`’ съдържа инструментите за зареждане на външни услуги и изпълнители от файлове в YAML формата, описан в 4.1.6. Всички услуги дефинирани в този пакет представляват двойки услуга-изпълнител. Пакетът съдържа и специалната услуга ‘`core`’.

Регистър на типове `untangle.types`. Пакетът ‘`untangle.types`’ съдържа реализацията на типовата система описана в 4.1.1. В него са реализирани всички операции между класове, дефинирани са стандартните типове, включващи числа (‘`Number`’, ‘`Integer`’, litReal и др.), текстови низове (‘`string`’), общи последователности (‘`Sequence`’), булеви стойности (‘`Boolean`’) и други. Включен е модулът ‘`prototype`’, който предлага инструментите за описание на прототипите на функции споменати в 4.1.2.1.

Работни потоци `untangle.workflow`. Пълната реализация на четеца и изпълнителя на работни потоци се намира в пакета ‘`untangle.workflow`’. Той включва пълната граматика на изразите като част от модула ‘`expression`’, която използва външния синтактичен LR(1) анализатор lrparsing.

Входни и изходни данни в пакета `format`. Пакетът ‘`format`’ съдържа инструментите за обработка на формати файлове и множества от данни. Те са разделени на две части – ‘`Data`’ класове, които описват типа данни и се регистрират в ‘`untangle.types`’, и ‘`Format`’ класове, които описват прочитането, записването и преобразуването на данните между различни файлови формати.

4.2.2 Инструменти за команден ред

Включени са два инструмента за команден ред – ‘`untangle-custom`’ и ‘`untangle-directory`’. Те служат съответно за единичното изпълнение на произволен работен поток, и за изпълнението на прост работен поток с един вход и изход върху директория съдържаща

повече от едно множества от данни.

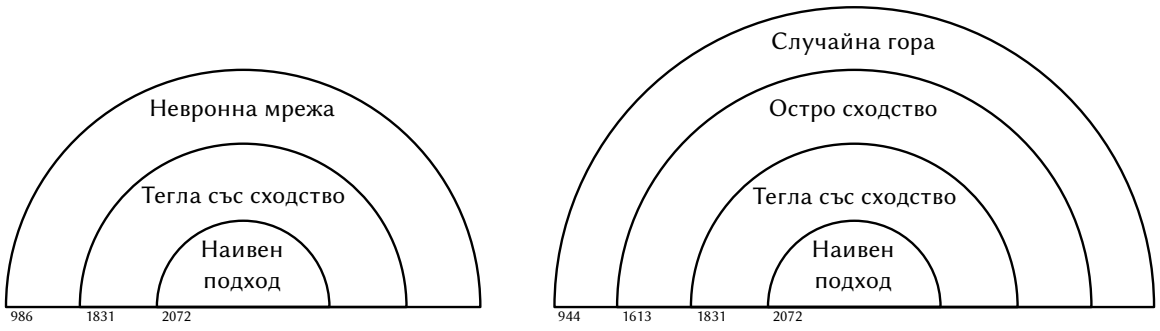
Untangle ще бъде публикуван като свободен софтуер под X11 лиценз[12], когато всички негови компоненти бъдат финализирани и интерфейсите му не подлежат на съществени промени.

[12] http://www.xfree86.org/3.3.6/COPYRIGHT2.html#3

Глава 5

Анализ на резултатите от проведените експерименти

За да бъдат оценени предложените в този труд подходи бяха проведени поредица от експерименти, целящи да потвърдят техните качества. Първоначално беше измерено разпределението на грешките, както за нуждите на симулатора на такива, така и за потвърждение на предварителните очаквания и избор на параметрите на методите. В следствие на това, че всеки от предложените методи бе прилаган под формата на разширение или подобрение на друг метод, в същинските експерименти бяха сравнени резултатите, постигнати със и без всяко отделно подобрение. Посочената градация на методи е илюстрирана на фигура 5.1, където се вижда как всеки следващ метод прави по-консервативен избор на потенциалните грешки.



(а) Сходство и невронна мрежа (б) Остро сходство и случайна гора

Фигура 5.1: Градация на методите

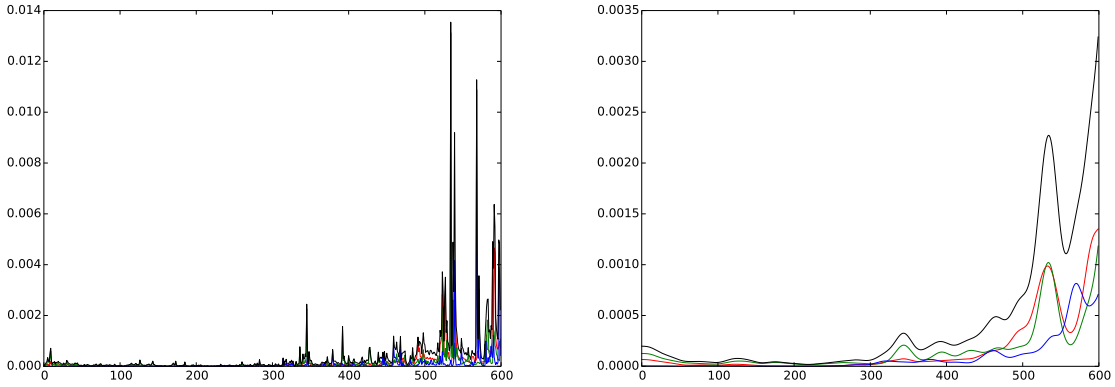
За невронните мрежи, случайните гори и останалите средства на изкуствения интелект бяха проведени отделни експерименти, оценяващи класификационната точност на използваните модели направо върху всички данни. Резултатите от тях бяха използвани за сравнение на различните модели и техните параметри, но крайната оценка беше получена едва след като избраните модели (невронни мрежи и случайни гори) бяха приложени върху примери за класификация предварително избрани с аналитичния подход, и отново беше оценено постигнатото подобрение в точността на класификация.

5.1 Изследване на разпределението на грешките

За да бъде използван методът за симулиране на грешки от 3.4.2.2, както и за нуждите на пресмятане на променлив праг, трябва количеството грешки в изследваните данни да бъде измерено. За тази цел с апарата за секвениране предварително беше секвенирано множества от данни, в които всички прочити са напълно идентични – общо 273217 прочита в 386 множества. Върху всяко едно от тях беше приложен наивния подход от 3.4.1.1 за корекция на грешките, след което беше преброен броя на грешките за всяка позиция. Резултата от това е показан на фигура 5.2а. На нея има три криви за трите вида грешки – заместания, инсерции и делеции. Най-горната четвърта крива показва сумата на трите.

След като бяха измерени честотите по позиция, върху тях беше приложен Гаусов заглаждащ филтър [60], резултата от който е показан на фигура 5.2б. Освен че прави разпределението по-прегледно визуално, заглаждането елиминира случайна вариация в честотата предизвикана от потенциален недостиг на прочити. Също така гарантира, че при използването му за плаващ праг няма да има съвпадение между симулираните грешки и прага, който би обезсмислил резултата от валидацията на такъв праг.

На графиките се забелязва, че е изпълнено очакването грешките да не са равномерно разпределени, и броят им силно да зависи от позицията [47]. Забелязва се силно засилване на броя грешки към края на прочитите, което е причина да съществува практиката те да бъдат подрязвани, когато са по-дълги от 300 бази.



(а) Действително

(б) След Гаусов заглаждащ филтър

Фигура 5.2: Разпределение на грешките

На графиките са показани само първите 600 бази, понеже те покриват обработваните тук данни. Броят грешки на по-високи позиции се засилва още повече – между позиции 800 и 900 броя грешки нараства до 40%. Това не представлява интерес за извършваните тук опити – толкова дълги прочити не бяха използвани, а и корекция на грешки при толкова много шум би била неприложима.

5.2 Експериментални резултати за аналитичния подход

5.2.1 Резултати от използване на сходството като тегло

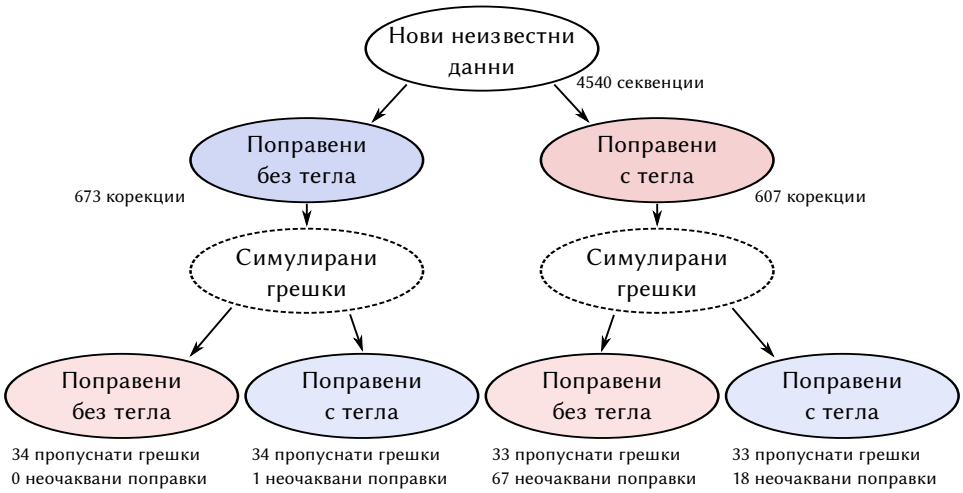
На тестов клъстер от 4540 секвенции, бяха приложени корекция с наивния подход (3.1) и претегления подход основаващ се на сходството (3.2) от 3.4.1 с помощта на емпирично избран праг за T^e.

С използване на наивния подход бяха направени 673 корекции, докато с използване на подобрения подход, използващ сходството, бяха направени 607, или с 66 помалко, което означава 10% намаление в броя на корекциите.

Бяха симулирани грешки чрез процедура 2 от 3.4.2.2 и грешките бяха коригирани отново (според процедура за валидация 3), при което не бяха отчетени разлики в пропуснатите симулирани грешки. Както се вижда на фигура 5.3, в множеството пред-

варително поправено с наивния подход, броят на пропуснати грешки е еднакъв (34) и за двата подхода, докато в множеството предварително поправено с този на сходство, броят на пропуснати грешки е отново еднакъв (33).

С други думи, въпреки че броят на корекции е намалял с 10%, което е значимо подобрение, броят на пропуснати симулирани грешки (които имат същото разпределение като действителните) не е намалял. Този резултат отговаря на очакването за намаляване на броя на излишни корекции, благодарение на оценката за сходство.



Фигура 5.3: Резултат от валидация чрез двукратна корекция

За пълнота, при повторната корекция на грешки беше оценен и броят нежелано поправени грешки. Трябва да се отбележи, че поради наличието на неизвестни грешки, качеството на поправяне на които е неясно, тази оценка не е обективна. Тя показва единствено броя грешки, които са били поправени, но не са били сред симулираните. Въпреки това тя би могла да участва при правенето на предположения за реалната бройка на нежелано поправените грешки.

Когато данните предварително бяха поправени с наивния подход, на втори етап нямаше почти никакви подобни корекции. Това отговаря на очакването, че още на първия етап наивния подход е поправил много грешки, които не са били грешки, и не е останало нищо, което той или другия подход да поправят на втория. За клона на данните предварително поправен с подхода на сходството, наивния подход отчита 67 допълнителни корекции, които отговарят на неговите 66 допълнителни замени

Ще бъдат представени множество сравнения между различни подходи с помощта на процедура 3 от 3.4.2.3. За по-лесно позициониране на текущото сравнение спрямо останалите, всяка таблица с такъв резултат ще бъде придружена от мини-таблица с отделните методи, в която сравняваните ще бъдат отбелязани. Всяка клетка отговаря на различен метод – наивен подход, аналитичен подход претеглен по сходство, аналитичен подход претеглен по остро сходство, както и техните комбинации с модел на изкуствена невронна мрежа и случайна гора. Тъй като аналитичните методи разчитат на прагове, клетките отговарящи на използване на фиксиран и променлив праг са отделени на два реда.

	Наив.	Сходство	Остро сх.	Сходство+ИНМ	Сходство+Случ. гора	Остро+ИНМ	Остро+СГ	SHREC
Фиксиран праг								
Променлив праг								

на първи етап, които са считани за излишни. В същото време претегленият подход води до само 18 нови поправки, които най-вероятно са нежелани, но със сигурност представляват понижена оценка за допуснатите нежелани поправки на първи етап.

За потвърждение на зависимостта от фигура 5.3 бяха проведени отделни експерименти върху различни множества от данни с различни настройки за прага.

Корекции				Изпуснати грешки			
Без тегла	Със тегла	Разлика		Без тегла	Със тегла	Разлика	
1548	1451	-97	(-6%)	392	391	-1	(-0.2%)
813	698	-115	(-14%)	386	391	+5	(+1.3%)
699	613	-86	(-12%)	415	412	-3	(-0.7%)
391	250	-141	(-35%)	445	448	+3	(+0.7%)
610	518	-92	(-15%)	70	69	-1	(-1.4%)
287	232	-55	(-19%)	73	74	+1	(+1.4%)
280	204	-76	(-27%)	81	82	+1	(+1.2%)

	Наив.	Сходство	Остро сх.	Сходство+ИНМ	Сходство+Случ. гора	Остро+ИНМ	Остро+СГ	SHREC
Фиксиран праг	◆	◆						
Променлив праг								

Таблица 5.1: Отношение на броя корекции и пропуснатите грешки

В таблица 5.1 са показани резултатите от изпълнението на тези потвърдителни експерименти. Във всеки един от тях се наблюдава намаляване на броя на корекции при използване на степента на сходство, и не се забелязва значително увеличаване

на броя на пропуснатите грешки в следствие от използването ѝ. От това може да се заключи, че използване на модификацията предложена в 3.4.1.2 води до предвидимо подобряване на получените резултати.

Тук броят на направените корекции не може да бъде сравнен директно с броя грешки, понеже не може да се направи добра прогноза за точния брой на последните. Относителни им честоти по позиция бяха оценени за целите на симулация на грешки, но действителния им брой зависи от множество други фактори, включително и от самите генетични последователности, които се секвенират [47].

В [47] измерената честота на грешките е 1.07%, но прочитите с които работим тук са минали през значително филтриране, включително елиминация на прекалено подългите, с което е премахнат региона в който се срещат най-много грешки. Освен това са елиминирани и много четения, за които оценката за качество на апарата за секвениране е била прекалено ниска. От друга страна, измерената от нас честота от 0.03% пък е била оценена върху проба, която е прекалено чиста и е възможно да има неестествено ниска честота на грешките. Затова подобряване на отношението пропуснати грешки към направени корекции е единствения критерий, по който може да бъде оценявано подобряването на качеството.

5.2.2 Резултати от прилагане на променлив праг

Тъй като грешките в данните не са равномерно разпределени, експериментите бяха повторени с променлив праг. Той беше изчислен на база на статистиките използвани в 3.4.2.2 за симулация на грешки, като за да се избегне съвпадение между симулираните грешки и прага, върху него беше приложен силен Гаусов заглаждащ филтър [60] по начина показан на фигура 5.2б.

В таблица 5.2 е показано сравнение на резултатите получени с вариране на прага на експериментални постановки с две тестови множества данни. Тъй като не може да бъде направено пряко съответствие между даден променлив праг и даден фиксиран праг, бяха направени по два отделни опита с различен фиксиран праг за един и същи променлив. По този начин най-явно се демонстрира от вариацията. Навсякъде е използван аналитичния подход претеглен с локалното сходство. Навсякъде промен-

Корекции				Изпуснати грешки			
Фиксиран	Променлив	Разлика		Фиксиран	Променлив	Разлика	
698	659	-39	(-5.59%)	534	467	-67	(-12.55%)
1451	659	-792	(-54.58%)	484	467	-17	(-3.18%)
232	177	-55	(-23.71%)	72	73	+1	(+1.39%)
518	177	-341	(-151.91%)	67	73	+6	(+8.22%)

	Наив.	Сходство	Остро сх.	Сходство+ИНМ	Сходство+Случ. гора	Остро+ИНМ	Остро+СГ	SHREC
Фиксиран праг		◆						
Променлив праг		◆						

Таблица 5.2: Подобрение при прилагане на променлив праг

ливия праг е изчислен като линейна функция от предсказаната честота на грешки.

Очаквано, при еднакво или леко по-добро количество на пропуснатите грешки се наблюдават по-малък брой корекции. Във всички опити използването на променлив праг доведе до подобряване на резултата с броя направени корекции. Зависимостта тук обаче не е толкова добре видима, понеже не могат да бъдат осигурени данни направени при едни и същи параметри, а и броя на направените корекции варира значително при вариране на прага, което допълнително затруднява прякото сравнение. Резултата обаче е достатъчно убедителен, поради което следващите експерименти с невронни мрежи и случайни гори са съчетани с аналитичния метод с плаващ праг.

5.2.3 Резултати от прилагане на острата мярка за сходство

В 3.4.1.3 беше предложена алтернативна функция за локално сходство (3.5) като подобрение на аналитичния подход. Тя беше наречена остра, понеже води до по-голямо отдалечаване на двойката прочити при всяка открита разлика между тях, което по-добре съответства на намаляването на вероятността тези прочити да са свързани. Беше извършено сравнение на резултата получен със стандартната функция за сходство (3.4) и острата функция (3.5), показано в таблица 5.3.

Както се забелязва, изборът на мярка на сходство има голямо значение за качеството на откриване на грешки. Резултата недвусмислено показва, че изборът на мярка са сходство, която отговаря по-точно на реалните очаквания за генетичното разстоя-

Корекции				Изпуснати грешки			
Стандартна	Остра	Разлика		Стандартна	Остра	Разлика	
850	818	-32	-3.765%	469	463	-6	-1.279%
				438	430	-8	-1.826%
250	225	-25	-10.000%	72	70	-2	-2.778%
				72	71	-1	-1.389%
1831	1613	-218	-11.906%	416	400	-16	-3.846%
				414	391	-23	-5.556%
663	584	-79	-11.916%	64	62	-2	-3.125%
				64	67	3	+4.688%
2177	2013	-164	-7.533%	402	380	-22	-5.473%
				379	371	-8	-2.111%
804	710	-94	-11.692%	49	47	-2	-4.082%
				53	52	-1	-1.887%

	Наив.	Сходство	Остро сх.	Сходство+ИНМ	Сходство+Случ. гора	Остро+ИНМ	Остро+СГ	SHREC
Фиксиран праг								
Променлив праг		◆	◆					

Таблица 5.3: Подобрение при прилагане на остра мярка за сходство

ние между двойката сравнявани прочити, води до значително по-добър резултат. Подобрението в резултата от използването на остра мярка за сходство се доближава по значимост до тази на въвеждането на самото сходство.

Разглеждане на други класове на функции на сходство, допълнително подобрение на предложените дотук, както и фина настройка на параметрите им предоставя една сериозна възможност за бъдещо развитие на предложените тук методи.

5.3 Приложение на невронни мрежи и други инструменти на ИИ

Много мощно средство за подобряване на резултата, получен дотук, е използването на машинно самообучение за изграждане на модели, които класифицират някакъв набор

от бази като правилни и погрешни. Моделът може да бъде използван върху набора от предсказани грешки от аналитичния подход с променлив праг за подобряване на резултата от него, или ако се окаже, че точността му е достатъчно висока – и върху изходните данни.

За да бъде възможно сравняването на резултати от повече от един алгоритъм за машинно самообучение, беше решено да бъде използвана софтуерна система, която поддържа голям набор от такива, и за целта беше избрана системата Weka [156]. С нея беше обучена изкуствена невронна мрежа, която беше тествана чрез деление на обучаващото множество, както и чрез друго множество, получено от различна биологична проба. Бяха направени опити да се варират параметрите и входът на мрежата, и беше направено сравнение с голям набор от други методи.

След това първоначално тестване, невронната мрежа беше приложена върху предсказанията от аналитичния подход получени в 5.2.1 и по същия начин беше валидирана със симулирани грешки имащи реалистично разпределение с процедурата от 3.4.2.3.

5.3.1 Резултати от обучение на ИНМ и други модели на ИИ

На първия етап от прилагането на модели на изкуствения интелект (ИНМ, случайни гори и други), те бяха обучени и тествани по специално генерирани множества по начина описан в 3.6.2. Върху извадка от входните данни бяха генерирани описаните в 3.6.1.2 статистики, които служат за вход на обучавания модел. Бяха симулирани виртуални грешки във всички възможни позиции на извадката, като за всеки пример от нея бяха генерирани два обучаващи примера – един реален, класифициран като правилен, и един с виртуална грешка, класифициран като погрешен.

На този етап различните модели изследвани сами за себе си с така получените множества, без да бъде използвана пълната процедура за валидация от 3. Вместо това, качеството на класификация на моделите беше тествано направо чрез деление на обучаващото множество, и върху данни получени от алтернативно множество. Това беше направено с цел първоначална оценка на параметрите на моделите, както и за да се придобие предварителна представа за качествата на различните модели за разглежданата задача. Тези предварителни резултати не отговарят непременно на при-

ложението на тези методи върху действителни данни, което е валидирано отделно в следващата секция.

5.3.1.1 Приложение на изкуствени невронни мрежи

Бяха обучени две невронна мрежи върху две множества с входни данни, които с изключение на извършеното съпоставяне не са били обработвани. След това те бяха тествани по три начина – чрез деление на обучаващото множество в съотношение 2 към 1, с тестване на друг отрязък от същото множество, след като то е било коригирано с аналитичния подход, и с два отрязъка от другото множество – един коригиран, и един необработен. В таблица 5.4 са показани получените резултати.

Обуч.	Тестово	Грешки	✓	✘	Точност	Верни	✓	✘	Точност
C1	C1 деление	19215	19154	61	99.682%	23063	22980	83	99.640%
C1	C1 кориг.	56458	55729	729	98.708%	68296	67962	334	99.510%
C1	C3 ориг.	50698	50323	375	99.260%	58688	58516	172	99.706%
C1	C3 кориг.	51131	50853	278	99.456%	59373	59145	228	99.615%
C3	C3 деление	17191	17162	29	99.831%	20000	19952	48	99.760%
C3	C3 кориг.	51131	50857	274	99.464%	59373	59186	187	99.685%
C3	C1 ориг.	56321	55940	381	99.323%	68025	67583	442	99.350%
C3	C1 кориг.	56458	56036	422	99.252%	68296	67879	417	99.389%
✓ – правилно класифицирани, ✘ – погрешно класифицирани									

Таблица 5.4: Опити с невронната мрежа

Вижда се, че двете обучени невронни мрежи правилно класифицират примерите от виртуални грешки с приемлива честота, и че това е в сила дори когато се използват тренировъчни данни получени от напълно отделно множество биологични данни. За съжаление резултатът е недостатъчно силен, за да може невронната мрежа да бъде приложена директно върху самите данни сама. Съотношението между сигнал и шум е много по-голямо от получената точност на откриване на правилни бази, въпреки че точността при откриване на грешките би била достатъчна. По тази причина беше решено ИНМ и другите модели на ИИ да бъдат комбинирани с аналитичния метод

от 3.4.1.2. Аналитичния подход ще бъде използван, за да бъдат избрани кандидати за грешки, които да бъдат класифицирани по-точно с обучения модел на ИИ.

Тук възниква въпросът дали съществува зависимост между резултата от така обучен модел на ИИ и резултата от аналитичния подход, който също е основан на честотите на срещане и сходството между прочитите. В най-добрия случай, такава зависимост би отсъствала и точността от таблица 5.4 би се запазила върху предварително подбраните кандидати от 5.2.1, което би свело броя неправилни корекции близо до нула. В най-лошия случай, такава зависимост би била пълна и резултата не би се променил. И двата варианта са прекалено нереалистични, поради което очакването е приложението на методи на ИИ върху така избрани кандидати да доведе до значително подобрение на класификацията им, но точността на модела на ИИ само върху тези кандидати също да спадне значително, оставяйки част от нежеланите корекции.

Също така в резултата може да се забележи, че точността се понижава видимо, когато теста се извърши върху различно множество. Това при избрания метод на обучение не е непременно проблем – виртуалните грешки са напълно различни и независими както от действителните, така и от симулираните. Това означава, че не е проблем да бъде обучена отделна невронна мрежа за конкретното множество от биологични данни – ако получения модел е прекалено зависим от характеристиките на биологичните данни, това не означава, че са нарушени качествата ѝ да открива различни грешки в същото множество – тези качества ще бъде валидирани отделно.

Същата зависимост обаче се забелязва и в данни получени от същото множество след корекция на грешките. Тъй като броя извършени корекции е едва 0.11%-0.15% от общия брой бази, то е изключително невероятно разликата в резултата да е в следствие от тези корекции. Това подсказва, че има поне някаква степен на прекомерно напасване към извадката от данните, което подсказва, че може би има нужда от корекция на параметрите, избора на метод или входните данни. Няколко възможни такива бяха изследвани, но тази зависимост при невронните мрежи остана.

5.3.1.2 Вариране на параметрите на алгоритъма

Резултатите в таблица 5.4 бяха постигнати при предварително зададените параметри – 30 входни неврона (10 бази радиус на прозореца), 16 скрити неврона. Това изглежда да

е разумен избор – прозорец с радиус от 10 бази дава относително добро покритие, което да отличи близките от далечните секвенции. Също избора на бройката на скрити неврони като половината от броя входни е често срещан начален избор, който се очаква да даде достатъчно добър резултат. Все пак е за предпочитане избора да се направи или поне провери емпирично, поради което бяха направени експерименти с различни вариации на параметрите. Всички опити са направени по два начина – върху деление на обучаващото множество, и върху отделно тестово множество.

Първоначално беше вариран радиуса на прозореца около оценяваната позиция (на всяка база от радиуса съответстват три входни неврона). В таблица 5.5 се виждат резултатите получени с радиуси от 1 до 6. Също бяха направени опити само с невроните отговарящи на външните 2, 4 и 6 бази от прозорец с радиус 10.

входните неврони,

	Тест чрез деление				Тест на други данни			
	Грешки		Правилни		Грешки		Правилни	
Радиус	✘	Точност	✘	Точност	✘	Точност	✘	Точност
10 (избрано)	84	99.562%	68	99.704%	456	99.095%	190	99.674%
6	64	99.666%	77	99.665%	390	99.226%	237	99.593%
5	74	99.614%	91	99.605%	403	99.200%	233	99.600%
4	75	99.609%	85	99.631%	408	99.190%	141	99.758%
3	76	99.604%	93	99.596%	239	99.525%	246	99.578%
2	99	99.484%	78	99.661%	341	99.323%	219	99.624%
1	186	99.031%	99	99.570%	480	99.047%	214	99.633%
външни 6	111	99.421%	153	99.336%	599	98.811%	381	99.347%
външни 4	166	99.135%	139	99.396%	790	98.432%	457	99.217%
външни 2	285	98.515%	151	99.344%	911	98.192%	348	99.403%
Деление: 19201 грешки, 23049 правилни бази Други данни: 50396 грешки, 58372 правилни бази								

Таблица 5.5: Вариране на броя входни неврони

Забелязва се, че при увеличаване на входните неврони, резултата се подобрява.

При радиус от 1 база, когато има само три входни неврона отговарящи за съседните две бази, той е чувствително по-слаб. Нормализира се при радиус от 2 бази, където 6 входни неврона обобщават данните от четворката съседни бази. Над радиус от 6 бази вече има насищане и подобрението е незначително, така че 18 входни неврона са напълно достатъчни. Макар и да се забелязва леко влошаване на точността на откриване на грешки в отделно тестово множество данни, няма достатъчно основание да се смята, че броя входни неврони води до прекомерно напасване – зависимостта от по-лош резултат на второто множество се забелязва и при по-малък брой неврони.

Много важно нещо, което може да се види в получените резултати е това, че невроните, отговарящи за по-отдалечени бази дават чувствително по-лош резултат. Така не само четирите най-близки бази се оказват достатъчни за точността, те са и необходими, за да бъде постигната. Това потвърждава, че *използването на локално сходство е разумен избор*, и че е добра идея то да бъде претеглено с отстоянието от разглежданата позиция.

В таблица 5.6 се вижда част от резултата от опитите за намаляване и увеличаване на броя скритите неврони. За провеждане на експеримента са използвани 30 входни неврона, като е проверено доколко е разумен избора на 16 скрити неврона.

Скрити неврони	Грешки	✓	✘	Точност	Правилни	✓	✘	Точност
Тест чрез деление								
16 (избрано)	19201	19117	84	99.562%	23049	22981	68	99.704%
17	19201	19116	85	99.557%	23049	22974	75	99.674%
15	19201	19124	77	99.598%	23049	22975	74	99.678%
Тест на други данни								
16 (избрано)	50396	49940	456	99.095%	58372	58182	190	99.674%
17	50396	49938	458	99.091%	58372	58172	200	99.657%
15	50396	49946	450	99.107%	58372	58160	212	99.636%

Таблица 5.6: Вариране на броя скрити неврони

Беше забелязано, че както се вижда на показаната част от резултатите, 16 е близо до оптималната стойност, и дори в 7 от 8 от ситуациите дава по-добър резултат от този

с 15 или 17 скрити неврона. Макар и това да не дава гаранции, че 16 е най-добрия избор, то е достатъчно близо до него, за да бъде използваната стойност при последваща валидация.

5.3.1.3 Нечислови промени на входа на невронната мрежа

Бяха направени и два други опити за промяна на характеристиките на входа.

Първия се отнася до подредбата на обучаващите примери. Прочитите от които те са генерирани се избират на случаен принцип, но при проведените дотук опити базите в тези прочити са взимани последователно. Беше направен опит в който беше проверен ефекта от това те да бъдат разбъркани по случаен начин, за да се елиминира възможността това да нарушава точността на обучените модели.

Втория опит се отнася до добавяне на нов вид входен неврон. Както беше споменато по-рано, прекаленото напасване към биологичните данни не е непременно нежелано. В същото време, в самите данни неравномерна биологическа вариация – започват със силно консервативен участък, и завършват с хипервариабилен такъв. Това може да повлияе на статистиките за честота в съответните участъци, и поради това модел, който го взима предвид, би могъл да донесе по-точен резултат.

В продължение на този труд това би могло да бъде извършено като се тренират отделни невронни мрежи за отделните участъци в прочитите. Тук обаче е направен опит да се прецени доколко това би могло да доведе до по-добър резултат като просто позицията в прочита е подадена като отделен входен неврон.

Резултатите от опита с разбъркване на входа и с добавяне на входен неврон за позицията са показани в таблица 5.7.

Разбъркването на данните подобрява резултата получен върху деление на обучаващото множество, но неочаквано го влошава при прилагане върху отделно тестово множество. Този резултат не е достатъчно категоричен за плюсовете и минусите от такава операция, но дава достатъчно основания същинските опити да бъдат проведени с модели, обучени без разбъркване.

Опитът с добавяне на неврон за входа на позицията е по-категоричен, в три от четири случая резултата получен с позиция е по-лош. Въпреки това, вземайки предвид казаното за това, че тестовете върху виртуални грешки не дават достатъчно добра

Промяна	Грешки	✓	✘	Точност	Правилни	✓	✘	Точност
Тест чрез деление								
Без промяна	19201	19117	84	99.562%	23049	22981	68	99.704%
Вход с позиция	19201	19127	74	99.614%	23049	22968	81	99.648%
Разбъркване	19112	19035	77	99.597%	23138	23068	70	99.697%
Тест на други данни								
Без промяна	50396	49940	456	99.095%	58372	58182	190	99.674%
Вход с позиция	50396	49855	541	98.926%	58372	58169	203	99.652%
Разбъркване	50396	49989	407	99.192%	58372	58159	213	99.635%

Таблица 5.7: Други промени в невронната мрежа

представа за прилагането на моделите върху кандидати за грешки, и за това беше взето решение обучените върху него модели да бъдат валидирани до край вместо да се разчита само на този предварителен резултат. Основната причина за този избор е, че значението на позицията е важно за бъдещото подобряване на резултата, и трябва много по-добро потвърждение на един негативен резултат с него.

5.3.1.4 Приложение на други модели на изкуствения интелект

Тръгвайки от предположението, че предложените в 3.6.1.2 статистики за вход на невронната мрежа могат да образуват лесно откриваеми прагове, които да могат да бъдат използвани за класификация на примерите, в 3.6.4 беше предложено да бъдат обучени дървета на решенията със същия вход като невронната мрежа.

За да се провери удачността на това предложение, бяха направени опити с по-голямата част моделите[1] използващи дървета на решенията включени в системата Weka [156]. В таблица 5.8 са показани резултатите от всички опити с дървета. Бяха направени опити и с останалите алгоритми, които са предлага Weka, но е включен единствено резултатът от алгоритъма RIPPER за обучение на правила, понеже резултатите постигнати с останалите бяха незадоволителни. Резултатите в таблицата са подредени

[1] В таблицата с резултатите всички алгоритми на които не е обърнато специално внимание в този труд са дадени с английските им имена, за да не се затруднява търсенето им.

по точност на класификация на правилните бази в отделно тестово множество – това помага при търсенето на алгоритъм, който би могъл да бъде приложен независимо, без допълване с аналитичния подход.

Получените резултати потвърждават очакването, че дърветата на решенията също са подходящ инструмент за тази задача. Голяма част от алгоритмите, използващи дървета, показват значително по-добър резултат от невронните мрежи. Това е още по-видимо, когато се провежда тест върху разделяне на обучаващото множество – което може да свидетелства за по-голямо напасване към данните при някои от алгоритмите за обучение.

Моделът на случайната гора дава най-добрия резултат от всички приложени, и това е постигнато във всеки един от случаите. Той е единственият, който постига точност, по-голяма от 99.9%, или по-малко от една грешка на 1000 бази. Това го прави най-добрия кандидат за независимо приложение без комбинация с аналитичния подход. Но дори при максималната точност от 99.95% постигната при опитите, броят на излишните корекции в данните би бил значително висок.

Моделът на случайна гора бе избран от всички да бъде валидиран с пълната процедура от 3.4.2.3 в допълнение към невронната мрежа. Трябва да се повтори, че добрият резултат, постигнат върху виртуалните грешки, не гарантира добър резултат при валидацията, затова невронната мрежа също участва в пълната валидация, въпреки привидно слабия си резултат. Освен това, тъй като двата метода са много различни по начина на действието си и водят до голяма разлика в резултатите, включването и на двата във валидацията може да донесе допълнителни интересни резултати.

5.3.2 Приложение на методи на ИИ върху кандидати, избрани с аналитичния подход

Обучените невронни мрежи и случайни гори бяха подложени на пълна валидация с вече използваната процедура от 3.4.2.3. При валидацията моделите не се използват направо, а комбинирани с аналитичния подход по начина, описан в 3.6.5 (и показан по-рано на фигура 3.7 на страница 92).

Невронната мрежа обучена с допълнителен входен неврон с позицията беше час-

	Тест чрез деление				Тест на други данни			
	Грешки		Правилни		Грешки		Правилни	
Подход	✘	Точност	✘	Точност	✘	Точност	✘	Точност
Alternating Decision Tree	110	99.427%	82	99.644%	140	99.722%	609	98.957%
Decision Stump	180	99.063%	107	99.536%	344	99.317%	334	99.428%
Logistic Model Tree	52	99.729%	48	99.792%	522	98.964%	329	99.436%
Naïve Bayes Tree	190	99.010%	69	99.701%	407	99.192%	325	99.443%
Ripple-Down Rule	74	99.615%	44	99.809%	351	99.304%	313	99.464%
LogitBoost Alt. Dec. Tree	72	99.625%	63	99.727%	238	99.528%	260	99.555%
Decision Tree + Naïve Bayes	99	99.484%	55	99.761%	351	99.304%	244	99.582%
Reduced-Error Pruning Tree	62	99.677%	55	99.761%	333	99.339%	242	99.585%
Support Vector Machine	78	99.594%	117	99.492%	245	99.514%	239	99.591%
Random Tree	60	99.688%	55	99.761%	555	98.899%	226	99.613%
Невронна мрежа	84	99.563%	68	99.705%	456	99.095%	190	99.675%
Best-First Tree	56	99.708%	52	99.774%	358	99.290%	150	99.743%
Functional Tree	66	99.656%	59	99.744%	455	99.097%	142	99.757%
Partial C4.5	54	99.719%	53	99.770%	477	99.053%	130	99.777%
Decision Table	111	99.422%	33	99.857%	732	98.548%	124	99.788%
Simple Cart	56	99.708%	51	99.779%	476	99.055%	105	99.820%
C4.5	62	99.677%	42	99.818%	499	99.010%	103	99.824%
Grafting C4.5	57	99.703%	44	99.809%	374	99.258%	89	99.848%
RIPPER Rule Learner	55	99.714%	58	99.748%	344	99.317%	74	99.873%
Случайна гора (10 дървета)	51	99.734%	23	99.900%	452	99.103%	34	99.942%
Случайна гора (60 дървета)	46	99.760%	18	99.922%	393	99.220%	29	99.950%
Деление: 19201 грешки, 23049 правилни бази Други данни: 50396 грешки, 58372 правилни бази								

Таблица 5.8: Опити с различни модели на машинно самообучение

тично валидирана с по-малко опити. Всяка мрежа беше валидирана върху множеството данни, от което беше избрана извадката с която мрежата бе обучена. Търсените

симулирани грешки нямат връзка с използваните при обучение виртуални грешки, поради което няма съвпадение между обучаващо и тестово множество, което да създава проблем с постигнатия резултат. Валидирана е способността на модела да класифицира грешки, които не са били известни при построяването му – грешките са били случайно симулирани след като е приключило обучението.

С други думи, тук е валидиран комбиниран подход за корекция на грешки, който следва следните стъпки:

1. Взима се подмножество от биологичните данни, описани в 3.2 или подобни.
2. Извършва се множествено съпоставяне на подмножеството.
3. Върху необработените съпоставени данни се обучава невронна мрежа или случайна гора, която да разпознава грешки.
4. Данните се коригират с аналитичния подход от 3.4.1.2.
5. От предложените с тях корекции се премахват всички, които невронната мрежа (или случайната гора) не класифицира като грешки.

В таблица 5.9 са показани резултатите от валидация на този подход приложен с невронна мрежа и случайна гора, и са сравнени с резултата, който се получава, когато се използва само аналитичния подход. Навсякъде аналитичният подход е използван с променлив праг, като върху различни множества данни са приложени различни прагове. Понеже процедурата от 3.4.2.3 е разклонена, на един резултат с брой на корекции съответстват двойка резултати с пропуснати грешки. Тъй като тук са сравнени четири метода, разклоненията са повече, но само две от тях са валидирани и показани в дясната половина на таблицата. Процентът на подобрение навсякъде е пресметнат спрямо наивния подход без тегла от 3.4.1.1, който е показан в първата колона на таблицата за сравнение.

От таблицата се вижда, че както невронната мрежа, така и случайната гора довеждат до двойно намаляване на броя на извършените корекции без значително влошаване на откритите грешки. Това е значително по-добър резултат от получените 10% при

Корекции							Изпуснати грешки					
Наив.	Сходство		ИНМ		Случайна гора		Наив.	Сходство		ИНМ		Случайна гора	
1908	1655	-13.2%	896	-53.0%	875	-54.1%	398	385	-3.3%	388	-2.5%	450	+13.1%
							406	388	-4.4%	392	-3.4%	450	+10.8%
723	581	-19.6%	372	-48.5%	385	-46.7%	70	67	-4.3%	69	-1.4%	67	-4.3%
							56	54	-3.5%	54	-3.5%	54	-3.5%
2374	2107	-11.2%	1132	-52.3%	1093	-53.9%	419	395	-5.7%	400	-4.5%	462	+10.3%
							353	332	-5.9%	336	-4.8%	378	+7.0%
892	754	-15.4%	470	-47.3%	478	-46.4%	49	47	-4.1%	47	-4.1%	48	-2.0%
							35	34	-2.8%	35	=	35	=
4534	3882	-14.3%	1827	-59.7%	1748	-61.4%	351	320	-8.8%	330	-6.0%	380	+8.3%
							338	309	-8.5%	315	-6.8%	363	+7.3%
1529	1347	-11.9%	798	-47.8%	785	-48.6%	40	39	-2.5%	41	+2.5%	40	=
							51	49	-3.9%	48	-5.8%	49	-3.9%

	Наив.	Сходство	Остро сх.	Сходство+ИНМ	Сходство+Случ. гора	Остро+ИНМ	Остро+СГ	SHREC
Фиксиран праг								
Променлив праг	◆	◆		◆	◆			

Таблица 5.9: Сравнение на резултата с използване на ИНМ и случайни гори

използване само на степен на сходство. Използването на сходство довежда до намаляване и на броя пропуснати грешки – зависимост, която се запазва и при използването на невронна мрежа.

Резултатът със случайната гора е малко по-спорен. Въпреки неочакваната точност получена на предварителните тестове с виртуални грешки, когато се прилага на практика не се забелязва същото преимущество над невронните мрежи. В някои от тестовете случайните гори дават по-добри резултати, но освен че в други това не е изпълнено, се забелязва силна вариация в броя на пропуснатите грешки. Качеството на резултата остава силно непредвидимо.

Освен това се забелязва и друга зависимост – във всички опити намаляване на броя корекции е за сметка на увеличаване на броя на пропуснатите грешки, и то значително. Тя допълнително подставя под въпрос качествата на модела на случайна гора, когато се прилага за тази задача.

Това също показва, че предположението, че не може да се разчита напълно на първоначалните резултати от предишната секция, е основателно. На действителния тест

извършен върху данни, които имат същите характеристики като реални данни, невронната мрежа се представя много по-добре.

Това несъответствие в резултата не е неочаквано, тъй като по-рано в 5.3.1.4 беше използвана голяма случайна извадка от всички данни, докато тук е използвано малко подмножество от данните, избрано по неслучаен критерий. Въпреки това остава интересен въпроса, каква е точната причина за него. Може да предположим, че случайната гора е по-сходна с аналитичния подход, като едно разумно обяснение за това е способността на ИНМ да открива много по-сложни зависимости в данните от тези, които откриват случайните гори, и от описаните с праговете използвани от аналитичния подход.

5.3.2.1 Комбинация с използване на неврон за позицията

В 5.3.1.3 беше обучена невронна мрежа, която има допълнителен неврон с позицията в прочита. Тя имаше за цел да включи вариацията в данните в процеса на обучение, но първоначалния резултат не беше обещават. В таблица 5.10 е показан резултатът от частичен тест, в който тази невронна мрежа беше комбинирана с аналитичния подход.

Корекции								Изпуснати грешки							
Наив.	ИНМ		ИНМ с позиция		Случайна гора			Наив.	ИНМ		ИНМ с позиция		Случайна гора		
2374	1132	-52.3%	1102	-53.6%	1093	-53.9%		419	400	-4.5%	406	-3.1%	462	+10.3%	
								353	336	-4.8%	338	-4.2%	378	+7.0%	
892	470	-47.3%	448	-49.8%	478	-46.4%		49	47	-4.1%	47	-4.1%	48	-2.0%	
								35	35	=	36	+2.9%	35	=	

	Наив.	Сходство	Остро сх.	Сходство+ИНМ	Сходство+Случ. гора	Остро+ИНМ	Остро+СГ	SHREC
Фиксиран праг								
Променлив праг	♦			♦ (със/без поз.)	♦			

Таблица 5.10: Подобрение при използване на позицията при обучение

Тук отново се потвърждава очакването, че предварителният резултат, получен по-рано, не е бил достатъчен за отхвърлянето на този модел. Както се вижда, поне в тези опити включването на позицията като вход намалява броя на корекциите, без да увеличава значително броя на пропуснатите грешки. Въпреки че и тук няма категоричен резултат в полза на единия или другия метод, резултатът дава достатъчно основание

в продължение на този труд да бъде изследван подход, използващ отделни модели, обучени с данни от отделни участъци на прочитите.

5.3.2.2 Комбинация с аналитичен метод с остро сходство

В резултатите показани дотук в таблици 5.9 и 5.10 методите на изкуствения интелект бяха комбинирани със стандартната мярка за сходство (3.4), която беше предложена първоначално в 3.4.1.2. Бяха направени отделни експерименти с острата мярка за сходство (3.5) от 3.4.1.3, които имат за цел да проверят дали значителното подобрение постигнато при прилагането ѝ в 5.2.3 се запазва след като метода бъде комбиниран с трите модела използващи ИИ. Напълно е възможно ползата от двата метода да се припокрива, и двата вида разстояние да имат много по-близко качество след комбинацията. На таблици 5.11а и 5.11б е показан резултата от това сравнение направено върху две отделни множества данни.

Тук изобщо не се наблюдава явно изразено предимство на която и да е мярка за сходство, като резултатите освен двусмислени са и изключително близки. Промяната на мярката за сходство е съизмерима с въвеждането на неврон за позицията в невронната мрежа, като положителния ефект от последния е много по-ясно видим. Това свидетелства, че при обучението си моделите на невронна мрежи и случайна гора са включили зависимости, които съответстват на по-острата мярка за сходство по един или друг начин.

Тъй като обаче подобна близост не беше забелязана, когато мярката за сходство беше премахната напълно, не може да считаме, че тя се припокрива напълно със зависимостите научени от ИНМ или случайната гора. Освен това е възможно част от преимуществото от острата мярка също да се запазва, но значителната промяна на мярката да нарушава възможността за сравнение при равен праг – за което би могла да е свидетелство нееднозначността на резултатите. На допълнително изследване подлежи и потенциала за допълнителна настройка на параметрите на острата мярка.

По тази причина този резултат не изключва напълно използването на комбинация на острата мярка за сходство с, например, ИНМ обучена с допълнителен неврон за позицията като възможност за бъдещо подобрение на процедурата по откриване на грешки.

ИИ алгоритъм	Корекции			Изпуснати грешки		
	Стандартно	Остро сходство		Стандартно	Остро сходство	
ИМН	986	998	+1.217%	423	407	-3.783%
				421	401	-4.751%
ИМН с позиция	959	966	+0.730%	422	406	-3.791%
				423	402	-4.965%
Случайна гора	946	944	-0.211%	487	470	-3.491%
				468	449	-4.060%

(а) Данни 1

ИИ алгоритъм	Корекции			Изпуснати грешки		
	Стандартно	Остро сходство		Стандартно	Остро сходство	
ИМН	422	427	+1.185%	64	62	-3.125%
				64	67	+4.688%
ИМН с позиция	401	411	+2.494%	66	65	-1.515%
				67	69	+2.985%
Случайна гора	432	435	+0.694%	64	62	-3.125%
				66	67	+1.515%

(б) Данни 2

	Наив.	Сходство	Остро сх.	Сходство+ИНМ	Сходство+Случ. гора	Остро+ИНМ	Остро+СГ	SHREC
Фиксиран праг								
Променлив праг				◆ (със/без поз.)	◆	◆ (със/без поз.)	◆	

Таблица 5.11: Ефект от комбиниране на остро сходство с изкуствен интелект

5.4 Сравнение със съществуващи решения

От разгледаните алтернативни методи за корекция на грешки, единствено V-Phaser [87] и SHREC [134] притежаваха качества, които да ги правят подходящи за задачата за откриване на грешки в метагеномни данни от паралелно de novo секвениране. V-Phaser използва статистически модел, който изследва съвместната вариация, който изглежда обещаващ и за разглежданата тук задача, тъй като представя друг начин за оценка на сходството. По подобен начин, при търсенето в суфиксните дървета SHREC имплицитно включва сходствата между прочитите.

Останалите разгледани методи не притежаваха свойства, които да оправдаят опити с тях. Coral [130] например разчита на много ниска вариация в данните, която не е

налична за използваните тук множества.

За съжаление не беше възможно да бъде изпълнен тест с V-Phaser, който по описанието си е най-обещаващият метод. Неговата целева насоченост е била секвенирането на генома на вируса на СПИН, и софтуерната му реализация очаква входни данни, подготвени за асемблиране на отделни геноми. Това прави невъзможно използваните тук данни да бъдат подадени като негов вход по приемлив начин без сериозни модификации на софтуера, или алтернативна реализация.

SHREC също има други очаквания към входните данни, но не беше трудно данните да бъдат преработени за нуждите му. Тъй като при използването на суфиксни дървета не се използва предварително съпоставяне, се наложи съпоставянето да бъде премахнато от данните и впоследствие да бъде възстановено в коригираните данни.

В таблица 5.12 е показан резултат от сравнение на SHREC с комбинацията от аналитичния подход и двата приложени тук метода на ИИ. Както в таблици 5.9 и 5.10, и тук са показани резултатите от една първоначална корекция, и последвалите две корекции на симулирани грешки в дясната част на таблицата.

Корекции					Изпуснати грешки				
SHREC	ИНМ		Случайна гора		SHREC	ИНМ		Случайна гора	
2321	1827	-21.284%	1748	-24.688%	374	330	-11.765%	380	+1.604%
					364	315	-13.462%	363	-0.275%
1370	920	-32.847%	897	-34.526%	52	52	=	52	=
					47	35	-25.532%	39	-17.021%

	Наив.	Сходство	Остро сх.	Сходство+ИНМ	Сходство+Случ. гора	Остро+ИНМ	Остро+СГ	SHREC
Фиксиран праг								◆
Променлив праг				◆	◆			

Таблица 5.12: Сравнение на комбинирания с ИИ метод и SHREC

Тъй като на SHREC не могат да се задават никакви съществени параметри се наложи да бъде променян прага на аналитичния подход, докато се стигне до сравним резултат, който е близък или по брой пропуснати грешки или по брой направени корекции. След изпълнение на SHREC беше търсен праг при който сравняваните алгоритми правят близък, но по-малък брой корекции, след което беше сравнен броя на пропуснатите грешки.

Както може да се види, при тези условия аналитичния метод, комбиниран с модел на ИИ, успява да постигне чувствително по-добър резултат при търсене на грешките от SHREC. При сходен или по-малък брой пропуснати грешки, SHREC прави чувствително повече корекции. При използването на невронни мрежи, такъв категоричен резултат беше успешно постигнат във всички изпробвани случаи.

Когато прага беше променен до достатъчно ниска стойност, дори след използването единствено на кандидатите за грешки получени само с аналитичния подход, без прилагането на ИНМ или случайни гори като допълнителен филтър, получения резултат беше сравним и често по-добър от постигнатия с SHREC. Това е показано в таблица 5.13 Показан е резултата със стандартната функция за сходство, както и със острата, с която има по-видимо преимущество над SHREC.

Корекции					Изпуснати грешки				
SHREC	Аналитичен подход		Остър подход		SHREC	Аналитичен подход		Остър подход	
2358	2177	-7.676%	2013	-14.631%	386	402	+4.145%	380	-1.554%
					366	379	+3.552%	371	+1.366%
2345	2356	+0.469%	2207	-5.885%	403	388	-3.483%	375	-6.716%
					317	312	-1.577%	291	-8.202%
1357	1209	-10.906%	1060	-21.887%	48	48	=	44	=
					48	54	+12.500%	50	+4.167%
1369	1259	-8.035%	1104	-19.357%	42	46	+9.524%	42	=
					45	42	-6.667%	38	-15.556%

	Наив.	Сходство	Остро сх.	Сходство+ИНМ	Сходство+Случ. гора	Остро+ИНМ	Остро+СГ	SHREC
Фиксиран праг								◆
Променлив праг		◆	◆					

Таблица 5.13: Сравнение на аналитичния метод и SHREC

5.5 Приложение на разработените методи при изследване на генома на пшеницата

Анализът и секвенирането на генома на пшеницата [21] е задача, която прилича на разглежданата тук. Макар и да е задача за изследване на един геном, той е хексаплоиден – с три подгенома, които съдържат различни версии на различните гени. По тази

причина пречките при изследването му са близки до тези при изследване на проби, съдържащи повече от един организъм. От друга страна, тъй като има само три подгенома, потенциалът за прилагане на предложените тук подходи е дори още по-голям.

Версия на аналитичния подход беше приложена за търсене на точкови мутации в секвенционни данни от пшеницата. При търсенето на точкови мутации, което е задача, силно затруднена от шума внесен от гените от другите подгеноми, правилното откриване на грешките не е от толкова голямо значение. Достатъчно важен резултат би бил рекласифицирането на по-голяма част от потенциалните грешки като точкови мутации.

За задачата аналитичния подход бе модифициран да работи с плоска функция на разстояние, която използва равномерно тегло по цялата дължина на прочитите. Тази модификация бе въведена, понеже вариацията в данните е значително по-ниска и откриването на близки точки на вариация е по-малко вероятно.

След изпълнение на алгоритъма върху 101513585 бази от генома на пшеницата, броят на откритите грешки беше намален от 1838860 (1.81%) до 795299 (0.78%). Благодарение на това броят открити мутации беше увеличен от 1638598 (1.61%) на 2728285 (2.69%). Този резултат подлежи на бъдеща валидация, но съществува значителен потенциал за повишаване на качеството на класификация на точкови мутации при пшеницата с прилагане на предложените тук подходи. Допълването им с модел на невронни мрежи би усилило допълнително един такъв резултат.

5.6 Оценка на потенциала на размития индикатор на увереност

За оценката на директното приложение на размит индикатор на увереност при йерархично клъстериране бяха използвани примерни данни, за които са приложени резултатите от прилагането на претегленото разстояние на Хеминг H^f от 3.5.

При проведените опити, с въвеждането на размит индикатор на увереност всички изкуствено внесени грешки бяха правилно пренебрегнати при пресмятане на разстоянията. Както се очакваше, секвенциите с по-редки заместания (които се очаква да

са грешки) бяха оценени с по-малки разстояния, докато секвенциите с по-чести заместванията бяха поставени на по-големи разстояния. Това отговаря на очакванията, че по-честите заместванията са точкови мутации, а не грешки.

Във почти всеки случай, с внасянето на повече заместванията води до увеличаване на разстоянията – каквото е поведението на немодифицираните функции на разстояние H. Единственото изключение, при което се наблюдават изкривени резултати, е когато дадена секвенция се среща над 100 пъти по-често от останалите, и това се случва единствено, когато между двете групи има значително съответствие. Това е чувствително по-добър резултат от резултата постигнат с прилагане на наивната оценка за увереност (3.1), където подобни изкривявания са много по-често срещани.

В по-голямата част от случаите, резултатите съвпаднат за двете функции за сходство (3.4) и (3.5).

		(а)	(б)	(в)	(г)	(д)
(а)	ATGCGAT (10000)	.0000	.0031	**.0015**	.0062	.0110
(б)	AT*T*CGAT (100)	.0031	.0000	.0056	**.0022**	.0040
(в)	ATGC*T*AT (1)	**.0015**	.0056	.0000	.0041	.0072
(г)	AT*TC*T*A*T (100)	.0062	.0022	.0041	.0000	**.0001**
(д)	A*GTC*T*G*T (1)	.0110	.0040	.0072	**.0001**	.0000

Таблица 5.14: Редки заместванията и претеглени разстояния

В таблица 5.14 се виждат резултатите от експеримента, проведен с функцията за сходство (3.4) и пет групи секвенции. В първата колона са поставени самите кратки секвенции заедно с тяхната бройка, като разликите между тях са подчертани. Останалите колони показват разстоянията, пресметнати за всяка двойка. Секвенция (а) е еталон, (б) и (в) имат по едно заместване, но (в) е по-рядка, и е замислена да бъде сгрешена. Секвенция (г) съдържа и двете заместванията, докато (д) съдържа още две допълнителни и също е изключително рядка и замислена като сгрешена.

Размитият индикатор на увереност успешно понижава разстоянието до очакваната за погрешна (в), тъй като тя е много по-рядка в множеството от (б). Секвенциите (г) и (д) остават правилно идентифицирани като по-далечни от предходните две, и са

правилно клъстерирани заедно, докато в същото време са сметнати за по-близки до (б) отколкото до (в) – субституцията в (б) е по-сигурна от очакваната за погрешна (в).

Въпреки че резултатите са с функцията за разстояние (3.4), която има по-слаб ефект при увеличаване на различията, експоненциално намаляващата (3.5) доведе до сходен резултат при клъстерирането.

Един проблем, който беше забелязан е, че и при двата варианта, разстоянията бяха намалели значително, все едно размитият индикатор на увереност беше оценил всички заместявания като почти сигурни грешки. Това не би било правилно заключение, но не бяха намерени ситуации, в които води до нежелано изкривяване на клъстерирането. Явно дори в този си вид, размитият индикатор поставен директно в алгоритмите за обработка се превръща в алтернатива на другите подходи за справяне с грешките. Въпреки това повече работа в насока справянето с този и други проблеми е необходима.

5.7 Обобщение на резултатите

В направените експерименти, всички предложени подходи за откриване и корекция на грешки бяха приложени като надстройки на използването на честота на срещане на базите за определяне на тяхната достоверност. Беше показано, че отделните надстройки водят до повишаване на качеството на корекция изразяващо се в подобряване на оценката за пропуснатите грешки и направените корекции.

Като отправна точка за сравненията беше използвана обикновената честота на срещане, наричана тук наивен подход (3.4.1.1). Тя беше надстроена с въведената в този труд честота претеглена с функция за локално сходство (3.4.1.2), като беше показано че тя води до видимо подобрение в получените резултати. Беше показано, че използването на въведената в последствие остра функция за локално сходство (3.4.1.3) допълнително засилва ефекта от това подобрение.

Беше установено, че предложения тук подход за обучение на класификационни модели с инструменти на изкуствения интелект постига (3.6) изключително добра класификационна точност, при това с различни алгоритми за машинно самообучение. Бяха изследвани някои негови параметри и модификации, като в процеса на извърш-

ване на опитите бяха индиректно демонстрирани както нееднородността на данните, така и ползата от критерии за сходство, които са локализирани около изследваната позиция.

Поради изключителната несъразмерност на двете категории за класификация (правилно и погрешно прочетена база), се наложи класификационните апарати на невронните мрежи и на случайните гори също да бъдат приложени като надстройки над честотата за срещане под формата на допълнителен филтър след прилагане на аналитичния подход. Беше установено, че този допълнителен филтър довежда до още по-чувствително подобрение на получените резултати.

Предложения комбиниран подход беше сравнен със софтуера за корекция на грешки SHREC. Беше установено, че при използването на инструменти на ИИ в комбинация с аналитичния подход, резултатите са чувствително по-добри от получените с SHREC. В същото време беше показано, че дори при използване единствено на директните надстройки на честотата на срещане, без допълнително филтриране с помощта на ИИ, резултата остава сравним, на места и по-добър, но разликата беше значително по-неизразена.

Беше проверено, че въведените в този труд претеглени разстояния на Хеминг и Левенщайн използващи размит индикатор на увереност (3.5) притежават очакванията от тях свойства, което обосновава бъдещето им по-разширено изследване като инструмент за приложение на оценките за грешки в задълбочен анализ на данни без извършване на предварителна корекция.

Глава 6

Приноси и перспективи

Предложен е набор от средства за откриване и корекция на грешките в метагеномни данни от паралелно секвениране, с помощта на използване на инструментите на информатиката и изкуствения интелект. Качества на предложените средства са изследвани в детайли с помощта на разработения индиректен подход за валидация.

6.1 Теоретико-методични приноси на дисертационния труд

1. Разработен е аналитичен подход за корекция на грешки в метагеномни данни от паралелно секвениране. Той въвежда честота на срещане, претеглена по сходство, както и две различни мерки за сходство, локални за региона, изследван за потенциални грешки (3.4.1.2, 3.4.1.3).

2. Разработен е индиректен подход за валидация на различни методи за корекция на грешки върху данни от паралелно секвениране, в които грешките не са предварително известни. (3.4.2.3) За целите му е разработен и подход за симулация на грешки, който да бъде използван при процедурата на валидация (3.4.2.2).

3. Въведен е размит индикатор на увереност за приложение на оценка на грешките при анализ на секвенционни данни без корекция на грешките (3.5). С негова помощ са въведени претеглено разстояние на Хеминг H^f и претеглено разстояние

на Левенщайн L^f.

4. Предложен е модел на невронни мрежи, който да може да бъде обучен да класифицира потенциалните грешки с по-голяма точност от прякото приложение на аналитичния подход (3.6). Той е основан на разбиване на честотата на срещане в отделни групи по позиционно сходство (3.6.1.2).

5. Предложен е подход за обучение на модели на машинното самообучение за откриване на грешки без да са налични обучаващи данни с действителни грешки. Той използва виртуални грешки на всички разглеждани позиции (5.3.1). Предложеният подход е независим от данните и по начина си приложение наподобява самообучение без учител, което предразполага към пълната автоматизация на обучението и приложението му.

6. Създаден е модел на система за извършване на сложен анализ на секвенционни данни, като е въведен език за описание на управляеми работни потоци, който е реализиран в софтуерния пакет untangle (4.1).

6.2 Експериментални приноси на дисертационния труд

1. Постигнато е експериментално потвърждение на ползата от въведената претеглена честота на сходство (5.2.1).

2. Постигнато е експериментално потвърждение на ползата от прилагане на променлив праг при откриване на грешките с честоти (5.2.2).

3. Обучени са невронна мрежа и 19 други модела на машинното самообучение. Извършен е експериментален анализ на възможните параметри на невронната мрежа (5.3.1.2), на потенциала на другите модели (5.3.1.4), както и на някои възможности за подобрение. В процеса на анализ е потвърдена ползата от прилагане на локална мярка за сходство (5.3.1.2).

4. Постигнато е експериментално потвърждение на ползата от допълване на аналитичния подход за корекция на грешки с модел на невронна мрежа или случайна гора (5.3.2).

5. Постигнати са експериментални резултати, потвърждаващи потенциала за използване на предложените методи за други задачи като откриване на точкови мутации в генома на пшеницата (5.5).

Списък с авторски разработки и предложения

1. Претеглена честота на срещане. (3.4.1.2)
2. Локална мярка за сходство на символни низове, претеглена с отстоянието от дадена позиция. (3.4.1.2)
3. Индиректен метод за валидация, използващ многократна корекция и размити множества. (3.4.2.3)
4. Симулатор на грешки. (3.4.2.2)
5. Претеглено разстояние на Хеминг и претеглено разстояние на Левенщайн. (3.5)
6. Предложение за разбиване на честотата на срещане по позиционно сходство за обучение на невронни мрежи. (3.6.1.2)
7. Предложение за използване на виртуални грешки за обучение на невронни мрежи при тяхната класификация. (5.3.1)

6.3 Перспективи за бъдещо развитие

Представените в този труд методи за откриване и корекция на грешки представляват добра основа за бъдеща работа, в която да се обхване по-широк кръг от задачи, както и да се навлезе в по-задълбочени изчислителни изследвания.

1. Разгледаните методи имат потенциала да бъдат приложени за класификационни задачи за откриване на структурни варианти като точкови мутации. Откритите вариации, които след използването на допълнителните подобрения (сходство, променлив праг или невронни мрежи) са рекласифицирани като правилни бази, са кандидати за точкови мутации. При доближаване на броя на пропуснати

грешки до 0, било то и с цената на голям брой нежелани корекции, останалите вариации могат да бъдат категорично определени като точкови мутации.

2. Разгледаните методи могат да намерят приложението си към данни, различни от метагеномните. Добър пример е геномът на пшеницата. При него усложненията, предизвикани от наличието на три подгенома, които съдържат различни версии на отделните гени, могат да бъдат разрешени с използването на методи, разчитащи на мерки за сходството.

3. Могат да се приложат алтернативни методи на валидация, основани на сравнения на филогенетични дървета. Например, може да бъде използвана оценката за правдоподобие на филогенетично дърво, построено по метода на максимално правдоподобие.

4. Може да се разработи симулатор на грешки, който използва много по-усъвършенстван статистически модел.

6.4 Декларация за оригиналност

Във връзка с провеждането на процедура за придобиване на образователната и научна степен „Доктор“ в Софийски университет „Св. Климент Охридски“ и защита на представения от мен дисертационен труд, декларирам, че:

- Резултатите и приносите на проведеното дисертационно изследване, представени в дисертационния труд на тема „Изкуствен интелект в биоинформатиката: автоматизиран анализ и класификация на данни от паралелно секвениране“, са оригинални и че те не са заимствани от изследвания и публикации, в които нямам участие.

- В текста на този дисертационен труд не са използвани неправомерно текстове и други обекти на авторското право без да бъде посочен източника, или без да съществува разрешение или законово право за това.

- Настоящата дисертация не е прилагана за придобиване на научна степен в друго висше училище, университет или научен институт.

- Представената от мен информация – списък с публикации, копия на документи и получени по време на експериментите резултати – отговаря на обективната истина.

6.5 Участия в проекти

Части от изследванията и разработките в този дисертационен труд са били спомогнати или извършени по време на моето участие в следните проекти:

- Проект на тема „Методи и информационни технологии за създаване, обединяване и използване на онтологии“ (договор № 177/2012 г. с Фонда за научни изследвания на Софийския университет)
- Проект на тема „Нови подходи и информационни технологии за създаване на специализирани системи, основани на знания“ (договор № 044/2014 г. с Фонда за научни изследвания на Софийския университет)
- Проект „Формиране на нова генерация от изследователи в областта на математиката, информатиката и компютърните науки чрез подкрепа на творческия и иновативен потенциал на докторанти, постодокторанти и млади учени във Факултета по математика и информатика на Софийски Университет „Св. Климент Охридски““ (договор № BG051PO001-3.3.06-0052 с МОМН с финансовата подкрепа на Оперативна програма „Развитие на човешките ресурси“, съфинасирана от Европейския социален фонд на Европейския съюз)

6.6 Публикации по темата на дисертационния труд

[C1] Milko Krachunov and Dimitar Vassilev. “An approach to a metagenomic data processing workflow”. In: *Journal of Computational Science* 5 (2014), pp. 357–362. DOI: 10.1016/j.jocs.2013.08.003.

[C2] Milko Krachunov. “Denoising of Metagenomic Data from High-Throughput Sequencing”. In: *Advanced Research in Mathematics and Computer Science*. Sofia, 2013, pp. 67–76.

[C3] Milko Krachunov. “Hierarchy and expressions for automated workflows for NGS data processing”. In: *Proceedings of the 8th International Conference on Information Systems & Grid Technologies (ISGT)*. [in press]. Sofia, May 30-31, 2014.

[C4] Milko Krachunov, Ognyan Kulev, Valeriya Simeonova, Maria Nisheva, and Dimitar Vassilev. “Manageable Workflows for Processing Parallel Sequencing Data”. In: *Serdica Journal of Computing* 8.1 (2014), pp. 1–14. ISSN: 1312-6555.

[C5] Milko Krachunov, Maria Nisheva, and Dimitar Vassilev. “Intelligent Approach for Automated Error Detection in Metagenomic Data from High-Throughput Sequencing”. In: *Proceedings of the 7th International Conference on Information Systems & Grid Technologies (ISGT)*. Sofia, June 31-1, 2013, pp. 160–168.

[C6] Milko Krachunov, Peter Petrov, Ivan Popov, and Dimitar Vassilev. “Computational challenges in a metagenomics processing pipeline”. In: *Proceedings of the 6th International Conference on Information Systems & Grid Technologies (ISGT)*. Sofia, June 1-2, 2012, pp. 302–311.

6.7 Доклади

[Д1] Milko Krachunov, Ivan Popov, Peter Petrov, Valeriya Simeonova, Maria Nisheva, Elena Todorovska, and Dimitar Vassilev. “Data analysis of next generation sequencing metagenomics studies - parallel computing approaches in genome assembly algorithms”. In: *Proceedings of regional conference “Supercomputing Applications in Science and Industry”*. Slanchev Briag, Bulgaria, Sept. 20-21, 2011.

[Д2] Milko Krachunov, Ivan Popov, Valeriya Simeonova, Irena Avdjieva, Pawel Szczęsny, Urszula Zelenkiewicz, Piotr Zelenkiewicz, and Dimitar Vassilev. “Detection and correction of errors in metagenomic 16S RNA parallel sequencing”. In: *Book of Abstracts of 4th workshop of EU COST action TD0801 Statistical challenges on the 1000€ genome sequences in plants (StatSEQ)*. Toulouse, France, Apr. 14-15, 2011.

6.8 Постери

[П1] Milko Krachunov and Dimitar Vassilev. “A weighted frequency approach for error detection in NGS metagenomic data”. In: *Book of abstracts of: 4th COST TD0801 “StatSEQ” Workshop*. Verona, Italy, Apr. 18-19, 2012.

[П2] Milko Krachunov and Dimitar Vassilev. “An approach to a metagenomic data processing workflow”. In: *Book of abstracts of: International Conference on Bioinformatics and Computational Biology BIOCOMP 2012*. Varna, Bulgaria, Sept. 20-22, 2012, p. 10.

[П3] Milko Krachunov and Dimitar Vassilev. “Intelligent Approach to Error Detection in Metagenomics Data from High-Throughput Sequencing”. In: *Book of abstracts of: 5th COST TD0801 “StatSEQ” Workshop*. Helsinki, Finland, Apr. 24-27, 2013.

[П4] Dimitar Vassilev, Milko Krachunov, Elena Todorovska, Valeriya Simeonova, Pawel Szczęsny, Pawel Siedlecki, Urszula Zelenkiewicz, and Piotr Zelenkiewicz. “Algorithm for error detection in metagonomics NGS data”. In: *Book of abstracts: COST BM1006 “NGS data analysis network” (SEQAhead) meeting in Brussels*. Nov. 7-9, 2011.

Литература

[1] Andrew Abbott and Angela Tsay. “Sequence Analysis and Optimal Matching Methods in Sociology: Review and Prospect”. In: *Sociological Methods and Research* 29.1 (2000), pp. 3–33. DOI: 10.1177/0049124100029001001.

[2] Stephen F. Altschul. “Amino acid substitution matrices from an information theoretic perspective”. In: *Journal of Molecular Biology* 219.3 (1991), pp. 555–565. DOI: 10.1016/0022-2836(91)90193-A.

[3] Matthew W. Anderson and Iris Schrijver. “Next Generation DNA Sequencing and the Future of Genomic Medicine”. In: *Genes* 1.1 (2010), pp. 38–69. ISSN: 2073-4425. DOI: 10.3390/genes1010038. URL: http://www.mdpi.com/2073-4425/1/1/38.

[4] The Austin Group. *make, The Open Group Base Specifications Issue 6, IEEE Std 1003.1*. 2004th ed. The IEEE and The Open Group. 2004. URL: http://pubs.opengroup.org/onlinepubs/009695399/utilities/make.html.

[5] Oswald T. Avery, Colin M. MacLeod, and Maclyn McCarty. “Studies on the Chemical Nature of the Substance Inducing Transformation of Pneumococcal Types: Induction of Transformation by a Desoxyribonucleic Acid Fraction Isolated from Pneumococcus Type III”. In: *The Journal of Experimental Medicine* 79.2 (Feb. 1, 1944), pp. 137–158. DOI: 10.1084/jem.79.2.137.

[6] Francisco J. Ayala. “Molecular clock mirages”. In: *Bioessays* 21.1 (Apr. 1999), pp. 71–75. DOI: 10.1002/(SICI)1521-1878(199901)21:1<71::AID-BIES9>3.0.CO;2-B.

[7] Ricardo A. Baeza-Yates and Gaston H. Gonnet. “Fast Text Searching for Regular Expressions or Automaton Searching on Tries”. In: *Journal of the ACM* 43.6 (1996), pp. 915–936. DOI: 10.1145/235809.235810.

[8] Mark W Barnett and Philip M Larkman. “The action potential”. In: *Practical Neurology* 7.3 (2007), pp. 192–197.

[9] Marina Barsky, Ulrike Stege, Alex Thomo, and Chris Upton. “A new method for indexing genomes using on-disk suffix trees”. In: *CIKM ’08: Proceedings of the 17th ACM Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2008, pp. 649–658. DOI: 10.1145/1458082.1458170.

[10] Nicholas H. Barton, Derek Briggs, Jonathan A. Eisen, David B. Goldstein, and Nipam H. Patel. “Models of Evolution”. In: *Evolution*. [Online chapter]. Cold Spring Harbor Laboratory Press, U.S., July 1, 2007. Chap. 28. URL: http://www.evolution-textbook.org/content/free/contents/ch28.html.

[11] Nicholas H. Barton, Derek Briggs, Jonathan A. Eisen, David B. Goldstein, and Nipam H. Patel. “Phylogenetic Reconstruction”. In: *Evolution*. [Online chapter]. Cold Spring Harbor Laboratory Press, U.S., July 1, 2007. Chap. 27. URL: http://www.evolution-textbook.org/content/free/contents/ch27.html.

[12] Serafim Batzoglou, David B. Jaffe, Ken Stanley, Jonathan Butler, Sante Gnerre, Evan Mauceli, Bonnie Berger, Jill P. Mesirov, et al. “ARACHNE: A Whole-Genome Shotgun Assembler”. In: *Genome Research* 12 (2002), pp. 177–189. DOI: 10.1101/gr.208902.

[13] Oren Ben-Kiki, Clark Evans, and Ingy döt Net. *YAML Ain’t Markup Language (YAML) Version 1.2, 3rd Edition*. [Online, accessed 18 December 2013]. Oct. 1, 2009. URL: http://yaml.org/spec/1.2/spec.html.

[14] Paul Bieganski, John Riedl, John Carlis, and Ernest F. Retzel. “Generalized Suffix Trees for Biological Sequence Data”. In: *Biotechnology Computing, Proceedings of the Twenty-Seventh Hawaii International Conference on*. 1994, pp. 35–44. DOI: 10.1109/HICSS.1994.323593.

[15] Frederick R. Blattner, Guy Plunkett, Craig A. Bloch, Nicole T. Perna, Valerie Burland, Monica Riley, Julio Collado-Vides, Jeremy D. Glasner, et al. “The complete genome sequence of Escherichia coli K-12”. In: *Science* 277.5331 (Sept. 1997), pp. 1453–1462. DOI: 10.1126/science.277.5331.1453.

[16] Mark Blaxter, Jenna Mann, Tom Chapman, Fran Thomas, Claire Whitton, Robin Floyd, and Eyualem Abebe. “Defining operational taxonomic units using DNA barcode data”. In: *Phil. Trans. R. Soc. B* 360.1462 (Oct. 2005), pp. 1935–1943. DOI: 10.1098/rstb.2005.1725.

[17] Rafal Bogacz, Malcolm W. Brown, and Christophe Giraud-Carrier. “Frequency-based error back-propagation in a cortical network”. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*. Vol. 2. Como, Italy, July 2000, pp. 211–216. URL: http://dml.cs.byu.edu/~cgc/pubs/IJCNN2000.pdf.

[18] Sébastien Boisvert, François Laviolette, and Jacques Corbeil. “Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies”. In: *Journal of Computational Biology* 17.11 (Oct. 2010), pp. 1519–1533. DOI: 10.1089/cmb.2009.0238.

[19] Plamenka Borovska, Veska Gancheva, and Soon-Heum Ko. “Scaling of Parallel Multiple Sequence Alignment on the Supercomputer JUQUEEN”. In: *The 7th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*. Berlin, Germany, Sept. 12-14, 2013.

[20] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (Oct. 1, 2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[21] Rachel Brenchley, Manuel Spannagl, Matthias Pfeifer, Gary LA Barker, Rosalinda D’Amore, Alexandra M Allen, Neil McKenzie, Melissa Kramer, et al. “Analysis of the bread wheat genome using whole-genome shotgun sequencing”. In: *Nature* 491.7426 (2012), pp. 705–710.

[22] Fiona S. L. Brinkman and Detlef D. Leipe. “Phylogenetic Analysis”. In: *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*. Ed. by Andreas D. Baxevanis and B.F. Francis Ouellette. Second. John Wiley & Sons, Inc., 2001. Chap. 14, pp. 323–358. ISBN: 0-471-38390-2. DOI: 10.1002/0471223921.ch14.

[23] William Brockman, Pablo Alvarez, Sarah Young, Manuel Garber, Georgia Giannoukos, William L Lee, Carsten Russ, Eric S Lander, et al. “Quality scores and SNP detection in sequencing-by-synthesis systems”. In: *Genome Research* 18.5 (2008), pp. 763–770.

[24] Nicolaas Govert de Bruijn. “A Combinatorial Problem”. In: *Koninklijke Nederlandse Akademie v. Wetenschappen* 49 (1946), pp. 758–764.

[25] Trevor Charles. “The Potential for Investigation of Plant-microbe Interactions Using Metagenomics Methods”. In: *Metagenomics: Theory, Methods and Applications*. Caister Academic Press, 2010. ISBN: 978-1-904455-54-7.

[26] Shalabh Chaturvedi. *Python Types and Objects*. [Online, accessed 27 Dec 2013]. 2009. URL: http://www.cafepy.com/article/python_types_and_objects/index.html.

[27] James Cheetham, Frank Dehne, Sylvain Pitre, Andrew Rau-Chaplin, and Peter J. Taillon. “Parallel CLUSTALW for PC Clusters”. In: *Proceedings of International Conference on Computational Science and Its Applications*. Vol. 2668. 2003, pp. 300–309. DOI: 10.1007/3-540-44843-8_32.

[28] Ramu Chenna, Hideaki Sugawara, Tadashi Koike, Rodrigo Lopez, Toby J. Gibson, Desmond G. Higgins, and Julie D. Thompson. “Multiple sequence alignment with the Clustal series of programs”. In: *Nucleic Acids Res* 31.13 (2003), pp. 3497–3500. DOI: 10.1093/nar/gkg500.

[29] Phillip E. C. Compeau, Pavel A. Pevzner, and Glenn Tesler. “How to apply de Bruijn graphs to genome assembly”. In: *Nature Biotechnology* 29.11 (Nov. 2011).

[30] George Cybenko. “Approximations by superpositions of sigmoidal functions”. In: *Mathematics of Control, Signals, and Systems* 2.4 (1989), pp. 303–314. DOI: 10.1007/BF02551274.

[31] Rolf Daniel. “The metagenomics of soil”. In: *Nat Rev Microbiol.* 3.6 (June 2005), pp. 470–478.

[32] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. 1st. London: John Murray, Nov. 24, 1859.

[33] Margaret O. Dayhoff, Robert M. Schwartz, and Bruce C. Orcutt. “A model of Evolutionary Change in Proteins”. In: *Atlas of protein sequence and structure*. supplement 3. Vol. 5. Nat. Biomed. Res. Found., 1978, pp. 345–358. ISBN: 0-912466-07-3.

[34] Narayan Desai, Dion Antonopoulos, Jack A. Gilbert, Elizabeth M. Glass, and Folker Meyer. "From genomics to metagenomics". In: *Current Opinion in Biotechnology* 23 (2012), pp. 72–76.

[35] Juliane C. Dohm, Claudio Lottaz, Tatiana Borodina, and Heinz Himmelbauer. "SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing". In: *Genome Research* 17.11 (Oct. 1, 2007), pp. 1697–1706. DOI: 10.1101/gr.6435207.

[36] Emmanuel J. P. Douzery, Frédéric Delsuc, Michael J. Stanhope, and Dorothée Huchon. "Local molecular clocks in three nuclear genes: divergence times for rodents and other mammals, and incompatibility among fossil calibrations". In: *Journal of Molecular Evolution* 57.1 (Aug. 2003), S201–S213. DOI: 10.1007/s00239-003-0028-x.

[37] John W. Drake, Brian Charlesworth, Deborah Charlesworth, and James F. Crow. "Rates of Spontaneous Mutation". In: *Genetics* 148.4 (Apr. 1, 1998), pp. 1667–1686.

[38] Robert C. Edgar. "MUSCLE: a multiple sequence alignment method with reduced time and space complexity." In: *BMC Bioinformatics* 5.1 (Aug. 19, 2004), p. 113. ISSN: 1471-2105. DOI: 10.1186/1471-2105-5-113.

[39] Michael Egmont-Petersen, Dick de Ridder, and Heinz Handels. "Image processing with neural networks—a review". In: *Pattern Recognition* 35.10 (2002), pp. 2279–2301. DOI: 10.1016/S0031-3203(01)00178-9.

[40] Belmont G. Farley and Wesley A. Clark. "Simulation of Self-Organizing Systems by Digital Computer". In: *IRE Transactions on Information Theory* 4.4 (1954), pp. 76–84. DOI: 10.1109/TIT.1954.1057468.

[41] Joseph Felsenstein. "Evolutionary trees from DNA sequences: A maximum likelihood approach". In: *Journal of Molecular Evolution* 17.6 (1981), pp. 368–376. DOI: 10.1007/BF01734359.

[42] Joseph Felsenstein. *Inferring Phylogenies*. Sunderland, Massachusetts: Sinauer Associates, 2004.

[43] Robert D. Fleischmann, Mark D. Adams, Owen White, Rebecca A. Clayton, Ewen F. Kirkness, Anthony R. Kerlavage, Carol J. Bult, Jean-Francois Tomb, et al. "Whole-Genome Random Sequencing and Assembly of Haemophilus Influenzae Rd". In: *Science* 269.5223 (1995), pp. 496–512. DOI: 10.1126/science.7542800.

[44] Paul Flicek and Ewan Birney. "Sense from sequence reads: methods for alignment and assembly". In: *Nature Methods Supplement* 6.11s (Nov. 2009), S6–S12. DOI: 10.1038/nmeth.1376.

[45] Robert W. Floyd. "Syntactic analysis and operator precedence". In: *Journal of the ACM* 10 (1963), pp. 316–333. DOI: doi:10.1145/321172.321179.

[46] Kimberly Gerrow and Antoine Triller. "Synaptic stability and plasticity in a floating world". In: *Current Opinion in Neurobiology* 20.5 (2010), pp. 631–639. DOI: 10.1016/j.conb.2010.06.010.

[47] André Gilles, Emese Meglécz, Nicolas Pech, Stéphanie Ferreira, Thibaut Malausa, and Jean-François Martin. "Accuracy and quality assessment of 454 GS-FLX Titanium pyrosequencing". In: *BMC Genomics* 12 (May 19, 2011), p. 245. DOI: doi:10.1186/1471-2164-12-245.

[48] Pascal Godefroit, Andrea Cau, Hu Dong-Yu, François Escuillié, Wu Wenhao, and Gareth Dyke. "A Jurassic avialan dinosaur from China resolves the early phylogenetic history of birds". In: *Nature* 498 (June 20, 2013), pp. 359–362. DOI: doi:10.1038/nature12168.

[49] Irving John Good. "Normal Recurring Decimals". In: *Journal of the London Mathematical Society* s3-21.3 (1946), pp. 167–169. DOI: 10.1112/jlms/s1-21.3.167.

[50] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech Recognition with Deep Recurrent Neural Networks". In: *The Computing Research Repository* abs/1303.5778 (Apr. 8, 2013). URL: http://arxiv.org/abs/1303.5778.

[51] Dan Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. U.S.A.: Cambridge University Press, 1997. ISBN: 0-521-58519-8.

[52] Peter Hamburg. "Fuzzy sets and de Morgan algebras". In: *Fuzzy Sets and Systems* 27.1 (1988), pp. 21–29.

[53] Donald Olding Hebb. *The Organization of Behavior*. New York: John Wiley & Sons, Inc., 1949.

[54] Steven Henikoff and Jorja G Henikoff. "Amino Acid Substitution Matrices from Protein Blocks". In: *PNAS* 89.22 (1992), pp. 10915–10919. DOI: 10.1073/pnas.89.22.10915.

[55] Kurt Hornik. "Approximation Capabilities of Multilayer Feedforward Networks". In: *Neural Networks* 4.2 (1991), pp. 251–257. DOI: 10.1016/0893-6080(91)90009-T.

[56] Kurt Hornik, Maxwell Stinchcombe, and Halber White. "Multilayer Feedforward Networks are Universal Approximators". In: *Neural Networks* 2 (1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

[57] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks". In: *Neural Networks* 3.5 (1990), pp. 551–560. DOI: 10.1016/0893-6080(90)90005-6.

[58] Shi Huang. "The genetic equidistance result of molecular evolution is independent of mutation rates". In: *Journal of Computer Science & Systems Biology* 1 (2008), pp. 92–102. DOI: 10.4172/jcsb.1000009.

[59] Weichun Huang, Leping Li, Jason R. Myers, and Gabor T. Marth. "ART: a next-generation sequencing read simulator". In: *Bioinformatics* 28.4 (Feb. 15, 2012), pp. 593–594. DOI: 10.1093/bioinformatics/btr708.

[60] Robert A. Hummel and David G. Lowe. "Computing gaussian blur". In: *Proceedings of the ICPR*. 1986, pp. 910–912.

[61] Chris I. Hunter, Alex Mitchell, Philip Jones, Craig McAnulla, Sebastien Pesseat, Maxim Scheremetjew, and Sarah Hunter. "Metagenomic analysis: the challenge of the data bonanza". In: *Briefings in Bioinformatics* 13.6 (Nov. 1, 2012), pp. 743–746. ISSN: 1477-4054. DOI: 10.1093/bib/bbs020.

[62] Susan M Huse, David Mark Welch, Hilary G Morrison, and Mitchell L Sogin. "Ironing out the wrinkles in the rare biosphere through improved OTU clustering". In: *Environmental microbiology* 12.7 (2010), pp. 1889–1898.

[63] Susan Huse, Julie Huber, Hilary Morrison, Mitchell Sogin, and David Welch. “Accuracy and quality of massively parallel DNA pyrosequencing”. In: *Genome Biology* 8.7 (2007), R143. ISSN: 1465-6906. DOI: 10.1186/gb-2007-8-7-r143. URL: http://genomebiology.com/2007/8/7/R143.

[64] Alec J. Jeffreys, Victoria Wilson, and Swee Lay Thein. “Hypervariable ’minisatellite’ regions in human DNA”. In: *Nature* 314 (Mar. 7, 1985), pp. 67–73. DOI: 10.1038/314067a0.

[65] Todd Jochem, Dean Pomerleau, Bala Kumar, and Jeremy Armstrong. “PANS: A Portable Navigation Platform”. In: *IEEE Intelligent Vehicles Symposium*. Detroit, Michigan, USA, 1995. URL: http://www.cs.cmu.edu/afs/cs/usr/tjochem/www/nhaa/navlab5_details.html.

[66] Mary Ellen Jones. “Albrecht Kossel, A Biographical Sketch”. In: *Yale Journal of Biology and Medicine* 26.1 (Sept. 1953), pp. 80–97.

[67] Kazutaka Katoh, Kei I. Kuma, Hiroyuki Toh, and Takashi Miyata. “MAFFT version 5: improvement in accuracy of multiple sequence alignment”. In: *Nucleid Acid Research* 33.2 (2005), pp. 511–518.

[68] Motoo Kimura. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, 1983. ISBN: 0-521-23109-4.

[69] Donald E. Knuth. “On the translation of languages from left to right”. In: *Information and Control* 8.6 (July 1965), pp. 607–639. DOI: doi:10.1016/S0019-9958(65)90426-2.

[70] *Albrecht Kossel - Biographical*. Physiology or Medicine 1901-1970, Elsevier Publishing Company, Amsterdam. 1967. URL: http://www.nobelprize.org/nobel_prizes/medicine/laureates/1910/kossel-bio.html.

[71] Albrecht Kossel. “Ueber Nucleïn der Hefe”. In: *Zeitschrift für physiologische Chemie* 3 (1879). (За нуклеина в маята), pp. 284–291.

[72] Albrecht Kossel. *Untersuchungen über die Nucleine und ihre Spaltungsprodukte*. (Изследвания върху нуклеина и продуктите му след деление). Strassburg, Germany: K.J. Trübne, 1881.

[73] David M. Kristensen, Arcady R. Mushegian, Valerian V. Dolja, and Eugene V. Koonin. "New dimensions of the virus world discovered through metagenomics". In: *Trends in Microbiology* 18.1 (Jan. 24, 2010), pp. 11–19. ISSN: 1878-4380. DOI: 10.1016/j.tim.2009.11.003.

[74] Victor Kunin, Anna Engelbrektson, Howard Ochman, and Philip Hugenholtz. "Wrinkles in the rare biosphere: pyrosequencing errors can lead to artificial inflation of diversity estimates". In: *Environmental microbiology* 12.1 (2010), pp. 118–123.

[75] Eric S. Lander, Lauren M. Linton, Bruce Birren, Chad Nusbaum, Michael C. Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, et al. "Initial sequencing and analysis of the human genome". In: *Nature* 409.6822 (Feb. 2001), pp. 860–921. DOI: 10.1038/35057062.

[76] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome". In: *Genome Biology* 10.3 (Mar. 4, 2009), 10:R25. DOI: 10.1186/gb-2009-10-3-r25.

[77] Robert W. Li, ed. *Metagenomics and its Applications in Agriculture, Biomedicine and Environmental Studies*. Nova Science Pub Inc, 2010.

[78] Ruiqiang Li, Hongmei Zhu, Jue Ruan, Wubin Qian, Xiaodong Fang, Zhongbin Shi, Yingrui Li, Shengting Li, et al. "De novo assembly of human genomes with massively parallel short read sequencing". In: *Genome Research* 20.2 (Feb. 2010), pp. 265–272. DOI: 10.1101/gr.097261.109.

[79] Weizhong Li, Limin Fu, Beifang Niu, Sitao Wu, and John Wooley. "Ultrafast clustering algorithms for metagenomic sequence analysis". In: *Briefings in Bioinformatics* 13.6 (Nov. 1, 2012), pp. 656–668. ISSN: 1477-4054. DOI: 10.1093/bib/bbs035.

[80] Weizhong Li and Adam Godzik. "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences". In: *Bioinformatics* 22.13 (July 1, 2006), pp. 1658–1659. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btl158.

[81] Zhenyu Li, Yanxiang Chen, Desheng Mu, Jianying Yuan, Yujian Shi, Hao Zhang, Jun Gan, Nan Li, et al. "Comparison of the two major classes of assembly algorithms: overlap–layout–consensus and de-bruijn-graph". In: *Briefings in Functional Genomics* 11.1 (2012), pp. 25–37. DOI: 10.1093/bfgp/elr035.

[82] Carolus Linnaeus. *Species Plantarum*. Stockholm: Laurentius Salvius, 1753.

[83] Carolus Linnaeus. *Systema Naturæ*. 10th. Stockholm: Laurentius Salvius, 1758.

[84] Charles Van Loan. *Computational Frameworks for the Fast Fourier Transform*. Frontiers in Applied Mathematics. SIAM, 1992. ISBN: 0-89871-285-8.

[85] Jan Łukasiewicz. "O logice trójwartościowej". In: *Ruch filozoficzny* 5 (1920). (За троичната логика), pp. 170–171.

[86] Chengwei Luo, Despina Tsementzi, Nikos Kyrpides, Timothy Read, and Konstantinos T Konstantinidis. "Direct comparisons of Illumina vs. Roche 454 sequencing technologies on the same microbial community DNA sample". In: *PloS ONE* 7.2 (2012), e30087.

[87] Alexander R. Macalalad, Michael C. Zody, Patrick Charlebois, Niall J. Lennon, Ruchi M. Newman, Christine M. Malboeuf, Elizabeth M. Ryan, Christian L. Boutwell, et al. "Highly sensitive and specific detection of rare variants in mixed viral populations from massively parallel sequence data". In: *PLoS Computational Biology* 8.3 (Mar. 15, 2012). DOI: 10.1371/journal.pcbi.1002417. URL: http://www.ploscompbiol.org/article/info:doi/10.1371/journal.pcbi.1002417.

[88] Mariette Manktelow. *History of Taxonomy*. Lecture from Dept. of Systematic Biology, Uppsala University. 2010. URL: atbi.eu/summerschool/files/summerschool/Manktelow_Syllabus.pdf.

[89] Emanuel Margoliash. "Primary structure and evolution of cytochrome C". In: *Proc. Natl. Acad. Sci.* 50.4 (Oct. 1963), pp. 672–679. DOI: 10.1073/pnas.50.4.672.

[90] Sally McBrearty and Nina G. Jablonski. "First fossil chimpanzee". In: *Nature* 437 (Sept. 1, 2005), pp. 105–108. DOI: 10.1038/nature04008.

[91] Warren McCulloch and Walter Pitts. "A Logical Calculus of Ideas Immanent in Nervous Activity". In: *Bulletin of Mathematical Biophysics* 5.4 (1943), pp. 115–133. DOI: 10.1007/BF02478259.

[92] Sara El-Metwally, Taher Hamza, Magdi Zakaria, and Mohamed Helmy. "Next-Generation Sequence Assembly: Four Stages of Data Processing and Computational Challenges". In: *PLoS Computational Biology* 9.12 (Dec. 12, 2013). DOI: 10.1371/journal.pcbi.1003345.

[93] Friedrich Miescher. “Ueber die chemische Zusammensetzung der Eiterzellen”. In: *Medicinisch-chemische Untersuchungen* 4 (1871). (За химическия състав на гнойните клетки), pp. 441–460.

[94] Dmitri Mikhailov, Haruna Cofer, and Roberto Gomperts. *Performance Optimization of Clustal W: Parallel Clustal W, HT Clustal, and MULTICLUSTAL*. Tech. rep. SGI Chem-Bio, Silicon Graphics, Inc, 2001.

[95] Jason R. Miller, Sergey Koren, and Granger Sutton. “Assembly Algorithms for Next-Generation Sequencing Data”. In: *Genomics* 95.6 (June 2010), pp. 315–327. DOI: 10.1016/j.ygeno.2010.03.001.

[96] Marvin Minsky and Seymour Papert. *Perceptrons: an introduction to computational geometry*. Cambridge, MA: The MIT Press, 1969. ISBN: 0-262-63022-2.

[97] Juan Antonio Morente-Molinera, Jose Manuel Martín, Carlos Cano, Marta Cuadros, and Armando Blanco. “SNP Annotation from Next Generation Sequencing Data”. In: *11th International Conference on Intelligent Systems Design and Applications*. 2011.

[98] Alison A. Motsinger-Reif and Marylyn D. Ritchie. “Neural networks for genetic epidemiology: past, present, and future”. In: *BioData Mining* 1.3 (17, 2008). DOI: 10.1186/1756-0381-1-3.

[99] David W. Mount. *Bioinformatics: Sequence and Genome Analysis*. second. Cold Spring Harbor Laboratory Press: Cold Spring Harbor, NY., 2004.

[100] Eugene W. Myers, Granger G. Sutton, Art L. Delcher, Ian M. Dew, Dan P. Fasulo, Michael J. Flanigan, Saul A. Kravitz, Clark M. Mobarry, et al. “A whole-genome assembly of Drosophila”. In: *Science* 287 (5461 Mar. 2000), pp. 2196–2204. DOI: 10.1126/science.287.5461.2196.

[101] Giuseppe Narzisi and Bud Mishra. “Comparing De Novo Genome Assembly: The Long and Short of It”. In: *PLoS ONE* 6.2 (Apr. 2011). DOI: 10.1371/journal.pone.0019175.

[102] Gonzalo Navarro. “A guided tour to approximate string matching”. In: *ACM Computing Surveys* 33.1 (2001), pp. 31–88. DOI: 10.1145/375360.375365.

[103] Sean Nee. “More than meets the eye”. In: *Nature* 429 (June 24, 2004), pp. 804–805. DOI: 10.1038/429804a.

[104] Saul B. Needleman and Christian D. Wunsch. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. In: *Journal of Molecular Biology* 48.3 (1970), pp. 443–453. DOI: 10.1016/0022-2836(70)90057-4.

[105] Karen E. Nelson and Bryan A. White. “Metagenomics and Its Applications to the Study of the Human Microbiome”. In: *Metagenomics: Theory, Methods and Applications*. Caister Academic Press, 2010. Chap. 10, pp. 171–182. ISBN: 978-1-904455-54-7.

[106] Cédric Notredame, Desmond G. Higgins, and Jaap Heringa. “T-Coffee: A novel method for fast and accurate multiple sequence alignment”. In: *J Mol Biol.* 302.1 (Sept. 8, 2000), pp. 205–217. DOI: 10.1006/jmbi.2000.4042.

[107] Howard Ochman and Allan C. Wilson. “Evolution in bacteria: evidence for a universal substitution rate in cellular genomes”. In: *Journal of Molecular Evolution* 26.1-2 (Nov. 1987), pp. 74–86. DOI: doi:10.1007/BF02111283.

[108] Joe Parker, Georgia Tsagkogeorga, James A. Cotton, Yuan Liu, Paolo Provero, Elia Stupka, and Stephen J. Rossiter. “Genome-wide signatures of convergent evolution in echolocating mammals”. In: *Nature* 502 (Oct. 2013), pp. 228–231. DOI: 10.1038/nature12511.

[109] Francis Jeffry Pelletier. “Metamathematics of Fuzzy Logic”. In: *The Bulletin of Symbolic Logic* 6.3 (Sept. 2000), pp. 342–346.

[110] Brant K. Peterson, Jesse N. Weber, Emily H. Kay, Heidi S. Fisher, and Hopi E. Hoekstra. “Double digest RADseq: an inexpensive method for de novo SNP discovery and genotyping in model and non-model species”. In: *PLoS ONE* 7.5 (2012). DOI: doi:10.1371/journal.pone.0037135.

[111] Erik Pettersson, Joakim Lundeberg, and Afshin Ahmadian. “Generations of sequencing technologies”. In: *Genomics* 93.2 (Feb. 2009), pp. 105–111. DOI: doi:10.1016/j.ygeno.2008.10.003.

[112] Pavel A. Pevzner, Haixu Tang, and Michael S. Waterman. “An Eulerian path approach to DNA fragment assembly”. In: *PNAS* 98.17 (2001), pp. 9748–9753. DOI: 10.1073/pnas.171285098.

[113] Sidney K. Pierce, Xiaodong Fang, Julie A. Schwartz, Xuanting Jiang, Wei Zhao, Nicholas E. Curtis, Kevin M. Kocot, Bicheng Yang, et al. "Transcriptomic Evidence for the Expression of Horizontally Transferred Algal Nuclear Genes in the Photosynthetic Sea Slug, Elysia chlorotica". In: *Molecular Biology and Evolution* 29.6 (2011), pp. 1545–1556. DOI: doi:10.1093/molbev/msr316.

[114] Hildete Prisco Pinheiroa, Aluísio de Souza Pinheiroa, and Pranab Kumar Sen. "Comparison of genomic sequences using the Hamming distance". In: *Journal of Statistical Planning and Inference* 130 (2005), pp. 325–329.

[115] Richard O. Prum. "Who's Your Daddy?" In: *Science* 322.5909 (Dec. 19, 2008), pp. 1799–1800. DOI: 10.1126/science.1168808.

[116] John Ross Quinlan. "Induction of Decision Trees". In: *Machine Learning* 1 (1986), pp. 81–106.

[117] Jens Reeder and Rob Knight. "Rapid denoising of pyrosequencing amplicon data: exploiting the rank-abundance distribution". In: *Nature methods* 7.9 (2010), p. 668.

[118] Siamak Rezaei and Md. Maruf Monwar. "Divide-and-Conquer Algorithm for Clustalw-MPI". In: *Proceedings of the Canadian Conference on Electrical and Computer Engineering, CCECE 2006, May 7-10, 2006, Ottawa Congress Centre, Ottawa, Canada*. 2006, pp. 717–720. DOI: 10.1109/CCECE.2006.277630.

[119] Martin Richards and Colin Whitby-Stevens. *BCPL — the language and its compiler*. Cambridge University Press, 1979. ISBN: 9780521219655.

[120] Daniel C Richter, Felix Ott, Alexander F Auch, Ramona Schmid, and Daniel H Huson. "MetaSim—A sequencing simulator for genomics and metagenomics". In: *PloS one* 3.10 (2008), e3373.

[121] Christian Rinke, Patrick Schwientek, Alexander Sczyrba, Natalia N. Ivanova, Iain J. Anderson, Jan-Fang Cheng, Aaron Darling, Stephanie Malfatti, et al. "Insights into the phylogeny and coding potential of microbial dark matter". In: *Nature* 499 (July 25, 2013), pp. 431–437. DOI: 10.1038/nature12352.

[122] Raul Rojas. *Neural networks: a systematic introduction*. Springer-Verlag, 1996. ISBN: 3-540-60505-3.

[123] Lior Rokach and Oded Maimon. *Data Mining with Decision Trees: Theory and Applications*. Series in Machine Perception and Artifical Intelligence. World Scientific Publishing Company, Apr. 1, 2008. ISBN: 978-9812771711.

[124] Mostafa Ronaghi, Mathias Uhlén, and Pål Nyrén. "A Sequencing Method Based on Real-Time Pyrophosphate". In: *Science* 281.5375 (July 17, 1998), pp. 363–365. DOI: 10.1126/science.281.5375.363.

[125] Frank Rosenblatt. "The Perceptron: A Probalistic Model For Information Storage And Organization In The Brain". In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519.

[126] Guido van Rossum. *Introducing Abstract Base Classes*. [PEP 3119]. Jan. 19, 2009. URL: http://www.python.org/dev/peps/pep-3119.

[127] David E. Rumelhart, James L. McClelland, and PDP Research Group. "Parallel Distributed Processing". In: vol. 1. The MIT Press, July 1986. Chap. 5. ISBN: 978-0-262181204.

[128] Camille Flye Sainte-Marie. "Question 48". In: *L'Intermédiaire Math.* 1 (1894), pp. 107–110.

[129] Naruya Saitou and Masatoshi Nei. "The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees". In: *Molecular Biology and Evolution* 4.4 (1987), pp. 406–425.

[130] Leena Salmela and Jan Schröder. "Correcting errors in short reads by multiple alignments". In: *Bioinformatics* 27.11 (June 1, 2011), pp. 1455–1461. DOI: 10.1093/bioinformatics/btr170.

[131] Steven L. Salzberg, Adam M. Phillippy, Aleksey Zimin, Daniela Puiu, Tanja Magoc, Sergey Koren, Todd J. Treangen, Michael C. Schatz, et al. "GAGE: A critical evaluation of genome assemblies and assembly algorithms". In: *Genome Research* 22.6 (Mar. 2012). DOI: 10.1101/gr.131383.111.

[132] Karsten Scheibye-Alsing, Steve Hoffmann, Annett M. Frankel, Peter Jensen, Peter F. Stadler, Yuan Mang, Niels Tommerup, Mike J Gilchrist, et al. "Sequence assembly". In: *Computational Biology and Chemistry* 33 (2009), pp. 122–136. DOI: 10.1016/j.compbiolchem.2008.11.003.

[133] Matthew B. Scholz, Chien-Chi Lo, and Patrick S. G. Chain. “Next generation sequencing and bioinformatic bottlenecks: the current state of metagenomic data analysis”. In: *Current Opinion in Biotechnology* 23 (2012), pp. 9–15.

[134] Jan Schröder, Heiko Schröder, Simon J Puglisi, Ranjan Sinha, and Bertil Schmidt. “SHREC: a short-read error correction method”. In: *Bioinformatics* 25.17 (2009), pp. 2157–2163.

[135] Jeffrey H. Schwartz and Bruno Maresca. “Do Molecular Clocks Run at All? A Critique of Molecular Systematics”. In: *Biological Theory* 1.4 (2006), pp. 357–371. DOI: 10.1162/biot.2006.1.4.357.

[136] Haixiang Shia, Bertil Schmidta, Weiguo Liua, and Wolfgang Müller-Wittiga. “Quality-score guided error correction for short-read sequencing data using CUDA”. In: *Procedia Computer Science* 1 (2010). DOI: 10.1016/j.procs.2010.04.125.

[137] Robert R. Sokal and Charles D. Michener. “A statistical method for evaluating systematic relationships”. In: *The University of Kansas science bulletin* 38.2 (1958), pp. 1409–1438.

[138] Lincoln Stein. “Genome annotation: from sequence to biology”. In: *Nature Reviews Genetics* 2.7 (2001), pp. 493–503. DOI: doi:10.1038/35080529.

[139] Gregor Stiglic, Simon Kocbek, Igor Pernek, and Peter Kokol. “Comprehensive Decision Tree Models in Bioinformatics”. In: *PLoS ONE* (Mar. 30, 2012). Ed. by Ahmed Moustafa, e33812. DOI: 10.1371/journal.pone.0033812.

[140] Greg Stuarta, Nelson Sprustonb, Bert Sakmannc, and Michael Häusserd. “Action potential initiation and backpropagation in neurons of the mammalian CNS”. In: *Trends in Neurosciences* 20.2 (Mar. 1, 1997), pp. 125–131. DOI: 10.1016/S0166-2236(96)10075-8.

[141] Sing-Hoi Sze, Yue Lu, and Qingwu Yang. “A polynomial time solvable formulation of multiple sequence alignment”. In: *J Comput Biol* 13.2 (2006), pp. 309–319. DOI: 10.1089/cmb.2006.13.309.

[142] Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice". In: *Nucleic Acids Res* 22.22 (1994), pp. 4673–4680. DOI: 10.1093/nar/22.22.4673.

[143] Tracy Tucker, Marco Marra, and Jan M. Friedman. "Massively Parallel Sequencing: The Next Big Thing in Genetic Medicine". In: *Am J Hum Genet* 85.2 (Aug. 14, 2009), pp. 142–154. DOI: 10.1016/j.ajhg.2009.06.022.

[144] José R. Valverde and Rafael P. Mellado. "Analysis of Metagenomic Data Containing High Biodiversity Levels". In: *PLoS ONE* 8.3 (2013). DOI: 10.1371/journal.pone.0058118. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3591453/.

[145] Martin Vingron and Michael S. Waterman. "Sequence alignment and penalty choice: Review of concepts, case studies and implications". In: *Journal of Molecular Biology* 235.1 (1994), pp. 1–12. ISSN: 0022-2836. DOI: http://dx.doi.org/10.1016/S0022-2836(05)80006-3. URL: http://www.sciencedirect.com/science/article/pii/S0022283605800063.

[146] Karl V. Voelkerding, Shale A. Dames, and Jacob D. Durtschi. "Next Generation Sequencing: From Basic Research to Diagnostics". In: *Clinical Chemistry* 55.4 (Feb. 26, 2009), pp. 41–47. DOI: 10.1373/clinchem.2008.112789.

[147] Robert A. Wagner and Michael J. Fischer. "The string-to-string correction problem". In: *Journal of the ACM* 21.1 (1974), pp. 168–173. DOI: 10.1145/321796.321811.

[148] Lusheng Wang and Tao Jiang. "On the complexity of multiple sequence alignment". In: *J Comput Biol* 1.4 (1994), pp. 337–348. DOI: 10.1089/cmb.1994.1.337.

[149] René L. Warren, Granger G. Sutton, Steven J. M. Jones, and Robert A. Holt. "Assembling millions of short DNA sequences using SSAKE". In: *Bioinformatics* 23.4 (Feb. 2007), pp. 500–501. DOI: 10.1093/bioinformatics/btl629.

[150] William G. Weisburg, Susan M. Barns, Dale A. Pelletier, and And David J. Lane. "16S ribosomal DNA amplification for phylogenetic study". In: *J. Bacteriol.* 173.2 (Jan. 1991), pp. 697–703.

[151] Paul John Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (Oct. 1990), pp. 1550–1560. DOI: 10.1109/5.58337.

[152] Paul John Werbos. “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences”. PhD thesis. Harvard University, 1974.

[153] Paul John Werbos. *The Roots of Backpropagation: From ordered derivatives to neural networks and political forecasting*. Adaptive and learning systems for signal processing, communications, and control. New York: John Wiley & Sons, Inc., 1994. ISBN: 0-471-59897-6.

[154] David R. Westhead, J. Howard Parish, and Richard M. Twyman. *Bioinformatics*. Instant Notes. BIOS Scientific Publishers Ltd., 2002. ISBN: 1-85996-272-6.

[155] Nathan R. Wilson, Caroline A. Runyan, Forea L. Wang, and Mriganka Sur. “Division and subt,raction by distinct cortical inhibitory networks in vivo”. In: *Nature* 488.7411 (Aug. 16, 2012), pp. 343–348. DOI: 10.1038/nature11347.

[156] Ian H. Witten, Eibe Frank, and Mark A. Hal. *Data Mining: Practical machine learning tools and techniques*. third. San Francisco: Morgan Kaufmann Publishers, 2011. ISBN: 978-0-12-374856-0.

[157] Carl R. Woese and George E. Fox. “Phylogenetic structure of the prokaryotic domain: the primary kingdoms”. In: *Proceedings of the National Academy of Sciences of the United States of America* 74.11 (Aug. 18, 1977), pp. 5088–5090. DOI: 10.1073/pnas.74.11.5088.

[158] John C. Wooley, Adam Godzik, and Iddo Friedberg. “A Primer on Metagenomics”. In: *PLoS Compt. Biol.* 6.2 (Feb. 26, 2010), pp. 289–290. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000667.

[159] Xing Xu, Hailu You, Kai Du, and Fenglu Han. “An Archaeopteryx-like theropod from China and the origin of Avialae”. In: *Nature* 475 (July 28, 2011), pp. 465–470. DOI: doi:10.1038/nature10288.

[160] Xiao Yang, Patrick Charlebois, Alex Macalalad, Matthew R. Henn, and Michael C. Zody. “V-Phaser 2: variant inference for viral populations”. In: *BMC Genomics* 14 (Oct. 3, 2013), p. 674. DOI: 10.1186/1471-2164-14-674.

[161] Jeffrey Yasskin. *A Type Hierarchy for Numbers*. [PEP 3141]. Jan. 21, 2010. URL: http://www.python.org/dev/peps/pep-3141.

[162] Lotfi A. Zadeh. "Fuzzy Sets". In: *Information and Control* 8 (1965), pp. 338–353.

[163] Moshe Zadka and Glyph Lefkowitz. "The Twisted Network Framework". In: *10th International Python Conference* (2002). URL: https://twistedmatrix.com/users/glyph/ipc10/paper.html.

[164] Daniel R Zerbino and Ewan Birney. "Velvet: algorithms for de novo short read assembly using de Bruijn graphs." In: *Genome Res.* 18.5 (2008), pp. 821–829. DOI: 10.1101/gr.074492.107.

[165] Emile Zuckerkandl and Linus Pauling. "Molecular disease, evolution, and genetic heterogeneity". In: *Horizons in Biochemistry*. New York: Academic Press, 1962, pp. 189–225.

[166] Marketa J. Zvelebil and Jeremy O. Baum. *Understanding Bioinformatics*. New York: Garland Science, 2008.

[167] Андрей Николаевич Колмогоров. "О представлении непрерывных функций нескольких переменных в виде суперпозиции непрерывных функций одного переменного и сложения". In: *Доклады Академий Наук СССР* 114.5 (1957), pp. 953–956.

[168] *Николай Кольцов*. [Online, accessed 12 Oct 2014]. URL: http://www.biology.ru/course/content/scientist/koltsov.html.

[169] Николай Константинович Кольцов. "Физико-химические основы морфологии". In: *Успехи экспериментальной биологии, сер. Б* 7.1 (1928), pp. 3–31.

[170] Владимир И. Левенштейн. "Двоичные коды с исправлением выпадений, вставок и замещений символов". In: *Доклады Академий Наук СССР* 163.4 (1964), pp. 845–848.

Списък на фигурите

2.1 Изкуствена невронна мрежа 28
2.2 Дърво на решенията 36
2.3 Прочит на ДНК последователност в цифров вид 38
2.4 Погрешно вмъкване при процеса на секвениране 41

3.1 Откъс от входните данни 65
3.2 Анализ на данни от метагеномно секвениране 66
3.3 Необходимост от съпоставяне 67
3.4 Сходство в прозорец около оценяваната позиция 70
3.5 Валидация чрез симулация на грешки и двукратна корекция 81
3.6 Примерен вход за невронната мрежа 88
3.7 Комбинация на аналитичния подход с невронна мрежа 92

4.1 Моделно изпълнение на работен поток 94
4.2 Примерен работен поток “quickalign” 103

5.1 Градация на методите 112
5.2 Разпределение на грешките 114
5.3 Резултат от валидация чрез двукратна корекция 115

Списък на таблиците

3.1 Поредица от множества с данни при неколкократна корекция 81

5.1 Отношение на броя корекции и пропуснатите грешки 116
5.2 Подобрение при прилагане на променлив праг 118
5.3 Подобрение при прилагане на остра мярка за сходство 119
5.4 Опити с невронната мрежа 121
5.5 Вариране на броя входни неврони 123
5.6 Вариране на броя скрити неврони 124
5.7 Други промени в невронната мрежа 126
5.8 Опити с различни модели на машинно самообучение 128
5.9 Сравнение на резултата с използване на ИНМ и случайни гори 130
5.10 Подобрение при използване на позицията при обучение 131
5.11 Ефект от комбиниране на остро сходство с изкуствен интелект 133
5.12 Сравнение на комбинирания с ИИ метод и SHREC 134
5.13 Сравнение на аналитичния метод и SHREC 135
5.14 Редки заместания и претеглени разстояния 137