Технически Университет – София

Филиал Пловдив

маг. Емилия Хаим Пардо

# РАЗПАРАЛЕЛЯВАНЕ НА АЛГОРИТЪМ NEEDLEMAN-WUNSCH ВЪРХУ ГРАФИЧНИ УСКОРИТЕЛИ

# АВТОРЕФЕРАТ

към дисертация за присъждане на научно-образователна степен “доктор”

Научен ръководител:

доц. д-р инж. Мария Маринова

Пловдив, 2025

# Съдържание

# Увод

Управлението на биологични данни представлява съществена предизвикателна задача, свързана със съхранението, индексирането, обработката и достъпа до големи хетерогенни масиви от данни. Използването на релационни и нерелационни бази данни (SQL/NoSQL), облачни изчислителни технологии, паралелни и разпределени изчисления позволява ефективна обработка на биоинформатични данни. Освен това, стандартизацията на файлови формати за биологична информация (например FASTA, BAM, VCF), отворените бази данни (NCBI, Ensembl, UniProt) и уеб услугите за обмен на данни ускоряват научните открития и улесняват глобалното сътрудничество.

Биоинформатиката разчита на разработването на високоефективни софтуерни алгоритми и инструменти, които намират приложение в геномното сглобяване, откриването на кодиращи последователности, структурната предикция на протеини, молекулното докиране и мрежовия анализ на биологични системи. Чрез интеграцията на изкуствен интелект (AI), биостатистика и изчислителни методи, биоинформатиката продължава да разширява границите на познанието в областта на биомедицинските науки и персонализираната медицина.

Ключов аспект при анализирането на генетични и протеинови данни е идентифицирането на функционални асоциации между гени или протеини, което изисква прилагане на алгоритми за множествено или двойно подравняване на последователности. Този процес позволява изчислително моделиране на еволюционните и функционални зависимости между биологичните секвенции чрез откриване на консервирани региони, мотиви и структурни особености.

## Основни цели и задачи на дисертацията

- Да се разгледа и анализира възможността за оптимизиране на алгоритъма Needleman-Wunsch чрез неговото изпълнение върху графичен ускорител (GPU) от типа NVIDIA, използвайки програмния модел CUDA C.

- Да се проучи специфичната архитектура на CUDA и как тя позволява едновременното изчисление на множество операции от различни нишки.
- Да се изследва и подчертае предизвикателството, свързано с прилагането на този подход към алгоритмите за динамично програмиране, където всяка клетка в матрицата зависи от предходните съседни клетки.
- Да се проучи паралелизация на алгоритъма Needleman-Wunsch, като методите за нея преодоляват това предизвикателство, като например метода с антидиагонален достъп до елементите на матрицата.
- Да се проведе експеримент как използването на GPU може да подобри производителността на алгоритъма Needleman-Wunsch чрез паралелна обработка.

# Глава 1

# Литературен обзор

## Структура и информационна функция на ДНК в контекста на компютърните науки

Дезоксирибонуклеиновата киселина (ДНК) представлява фундаментална биомолекула, която съдържа, съхранява и предава генетичната информация, необходима за изграждането и функционирането на живите организми [1]. От гледна точка на компютърната наука, тя може да бъде интерпретирана като биологична информационна система – база от данни, съхраняваща инструкции в линеен, последователен формат. Структурно ДНК е изградена от мономери – нуклеотиди, всеки от които включва фосфатна група, дезоксирибоза и една от четирите азотни бази (A, T, G, C). Нуклеотидите са свързани чрез фосфодиестерни връзки, формиращи захарно-фосфатен гръбнак, а комплементарните бази се сдвояват чрез водородни връзки, оформяйки двойната спирала – ключова триизмерна структура на молекулата.

В основата на ДНК се намира линейната последователност от азотни бази, наподобяваща двоична редица в цифровите технологии. От функционална перспектива, тази последователност определя кодирането на генетичната информация и нейното преобразуване в активни биомолекули чрез два последователни процеса – транскрипция и транслация. Транскрипцията представлява синтез на информационна РНК (иРНК), която служи като междинно представяне, аналогично на преобразуване на програмен код от един език в друг. Транслацията, от своя страна, интерпретира иРНК и синтезира съответните протеини – процес, съпоставим с компилиране на изпълним код от високо ниво.

Откритието на двойната спирала на ДНК от Уотсън и Крик през 1953 г. представлява крайъгълен камък в разбирането на молекулярната биология. Това откритие демонстрира как двойните вериги се свързват чрез комплементарно сдвояване (A-T, G-C), следвайки принципите на Уотсън-Крик и закона на Чаргаф, които гарантират стабилност и точност при репликация и експресия на генетичния код. В този смисъл, хромозомите, в които се организира ДНК, изпълняват ролята на структурирана и оптимизирана база данни, аналогична на йерархична файлова система в компютърните среди [2].

ДНК изпълнява функциите на изчислителна система чрез три основни процеса: **репликация**, при която информацията се дублира точно; **транскрипция**, при която се извлича и пренася междинен формат на информацията; и **транслация**, която реализира функционалния продукт – протеин. Всеки от тези процеси има своите биологични "алгоритми", включващи ензими и молекулни комплекси, действащи подобно на компилатори и интерпретатори в софтуерната архитектура.

Генетичният код функционира като формализирана комуникационна система – съставен от трибуквени кодони, които еднозначно дефинират аминокиселини. Тази система е едновременно детерминирана, редундантна и почти универсална – характеристики, които я правят устойчива и надеждна за предаване на информация. Началните и крайните кодони, например AUG (старт) и UAA/UAG/UGA (стоп), задават рамките на транслационния процес, подобно на синтактични конструкции в програмирането [3].

Накрая, съвременната биоинформатика – пресечна точка между молекулярната биология и компютърната наука – използва изчислителни модели, алгоритми и машинно обучение за анализ и оптимизация на геномни данни. Това позволява не само предсказване на биологични функции и заболявания, но и създаването на персонализирана медицина и генетични инженерни решения, с което ДНК се утвърждава като основа на една от най-сложните информационни системи, познати на науката.

## ДНК секвениране

ДНК секвенирането представлява процес на определяне на точната последователност на нуклеотидите в даден фрагмент от дезоксирибонуклеинова киселина. То има фундаментално значение за съвременната биология, тъй като позволява разчитането на генетичния код, заложен в четирите основни бази – аденин (A), тимин (T), гуанин (G) и цитозин (C). Чрез установяването на тази последователност се предоставя директен достъп до наследствената информация на организмите, като това открива широк спектър от приложения – от таксономично разграничаване на морфологично близки видове, до диагностика на генетични заболявания, откриване на патогенни агенти и използване в съдебната медицина [4].

ДНК секвенирането е от ключово значение и в геномиката и персонализираната медицина, където чрез анализ на индивидуалната генетична последователност могат да се предвидят предразположения към

заболявания, както и да се разработят таргетирани терапевтични подходи. Технологичният напредък в секвениращите платформи – от метода на Сангър до съвременните високопроизводителни (next-generation) техники – е катализирал революция в биомедицинските изследвания.

## Структура и функции на РНК

Рибонуклеиновата киселина (РНК) представлява линейна полимерна молекула, съставена от нуклеотиди, всеки от които включва рибоза, фосфатна група и една от четирите азотни бази – аденин (A), гуанин (G), цитозин (C) или урацил (U), който замества тимина в ДНК. За разлика от двойноверижната структура на ДНК, РНК обикновено е едноверижна, което ѝ придава структурна гъвкавост и способност да образува вторични и третични структури, включително клупове и стъбловидни участъци. Това свойство е от особено значение за функционалната диференциация на различните РНК молекули [5][6].

РНК изпълнява разнообразни биологични функции, които могат да се обособят в две основни категории: информационни и функционални. Информационната РНК (иРНК) служи като посредник между ДНК и процеса на протеинов синтез, пренасяйки копие от генетичната информация до рибозомите. Функционалните РНК включват рибозомната РНК (рРНК), основен структурен и каталитичен компонент на рибозомите, както и трансферната РНК (тРНК), която участва в интерпретацията на иРНК чрез доставяне на съответните аминокиселини. Други класове, като микроРНК и малки интерферентни РНК, играят критична роля в регулацията на генната експресия и в защитата срещу вирусни елементи.

## Химична стабилност и молекулярна еволюция на РНК

РНК молекулите, макар и по-малко стабилни от ДНК, притежават уникални химични свойства. Наличието на хидроксилна група на 2'-позиция на рибозата прави РНК по-реактивна и податлива на хидролиза, но също така ѝ придава каталитичен потенциал. Това е от особено значение за рибозимите – РНК молекули с ензимна активност, които демонстрират, че РНК може да бъде не само преносител на информация, но и катализатор. Така възниква хипотезата за "РНК-свят" в ранната еволюция, според която РНК е изпълнявала както информационни, така и функционални роли преди появата на ДНК и протеините [7].

## Основна догма и универсалност на информационния поток

Основната догма в молекулярната биология описва потока на генетичната информация от ДНК към РНК и накрая – към протеините. Този централен принцип обобщава трансформацията на наследствената информация в биологична функция чрез транскрипция и транслация. Въпреки че догмата се смята за универсална, при определени вируси (напр. ретровируси), РНК играе роля на генетичен материал, който може да бъде обратнопреписан в ДНК. Това демонстрира гъвкавостта на РНК и нейното централно значение в генетичната регулация и еволюция.

В обобщение, както ДНК секвенирането, така и разбирането на структурата и функциите на РНК са съществени за развитието на молекулярната биология, биотехнологиите и медицината. Те разкриват фундаментални механизми за управление на биологичната информация, като в съвременния контекст все по-често се разглеждат чрез аналогии с информационните технологии и компютърните науки.

## Алгоритми за подравняване на секвенции

Подравняването на ДНК или протеинови секвенции е основен метод в биоинформатиката, използван за сравнение на хомоложни региони между различни биологични последователности. То се третира като оптимизационен проблем, при който се търси най-доброто съвпадение между символите на секвенциите чрез минимизиране на вмъквания, делеции и замени. Различават се два основни типа подравняване: **по двойки** (глобално и локално) и **множествено подравняване**, което включва три или повече секвенции. Глобалното подравняване (напр. чрез алгоритъма на Needleman-Wunsch [8][9]) се прилага при сходни по дължина секвенции и цели изчерпателно сравнение. Локалното подравняване (напр. чрез Smith-Waterman[10][11][12]) е подходящо за откриване на съвпадения в частични региони [13]. Множественото подравняване (MSA) анализира структурни и функционални мотиви между много секвенции, с приложения в еволюционен анализ и структурно предсказване. Алгоритми като Dot Matrix [14] и методите на динамично програмиране се използват за изчисление на сходство, като се основават на скорингови матрици и backtracking. Изборът на метод зависи от контекста, целите и изчислителните ресурси.

## CUDA технологията в биоинформатиката

CUDA технологията, разработена от NVIDIA, представлява паралелна изчислителна архитектура, която използва графични процесори (GPU) за общо

предназначение (GPGPU) с цел значително ускоряване на изчислителни задачи. В биоинформатиката тя намира широко приложение благодарение на способността си да обработва големи обеми от данни паралелно, което я прави изключително ефективна за анализ на геномни данни, молекулно моделиране и симулации на биологични процеси. CUDA позволява използването на познати програмни езици като C/C++ и предлага програмни абстракции за управление на нишки, памет и синхронизация, осигурявайки мащабируемост и висока производителност. Чрез оптимално използване на различни типове памет – глобална, споделена, локална и текстурна – разработчиците могат да създават изключително ефективен код, подходящ за нуждите на съвременната биоинформатика.

**Алгоритми за подравняване на секвенции**

Алгоритмите за подравняване на секвенции се класифицират на два основни вида: **по двойки** и **множествени подравнявания**. При по-двойковите методи се използват техники като **динамично програмиране** (напр. алгоритмите на Needleman-Wunsch [16][17][32][33] и Smith-Waterman [15][28]) за намиране на глобално или локално оптимално подравняване, като за големи геноми се прилагат по-ефективни методи като MUMmer, MSA и SSAHA[29][30][31]. Процесът обикновено включва откриване на консервативни региони (котви), подравняване на непокриващи се котви и запълване на празнини.

Множествените подравнявания се делят на глобално-оптимизиращи, приблизителни, евристични (прогресивни и итеративни) [18][19] и вероятностни методи. Последните използват модели като скрити марковски модели (HMM). Евристичните методи като ClustalW, MAFFT и MUSCLE са широко използвани заради баланса между точност и ефективност. Съществуват и **комбинирани** подходи (напр. MCoffee), както и методи, подпомагани от бази данни или структурна информация (напр. 3DCoffee, PSI-PRALINE). За **локално множествено подравняване** се използват инструменти като **diAlign**, **ProDa** и **EulerAlign** [34-52].

# Глава 2

# CUDA архитектура и паралелно програмиране

## Въведение в паралелното програмиране

Въведение в паралелното програмиране обхваща принципите на едновременното изпълнение на програмни инструкции, като надгражда традиционния последователен модел на изпълнение. Всяка програма, стартирана на компютър, се изпълнява като процес, представляващ инстанция на програма с отделно адресно пространство и ресурси. При многозадачна среда множество процеси се управляват чрез механизми като кръгова опашка и приоритетно планиране. При паралелното програмиране се въвежда понятието *нишка* – лека единица на изпълнение, която споделя ресурсите на процеса. Основно предимство на паралелизма е възможността различни части от програмата да се изпълняват едновременно, без взаимни зависимости, което води до значително ускоряване на изчисленията.

**CUDA** (Compute Unified Device Architecture) е паралелна изчислителна платформа на NVIDIA, която използва графичния процесор за изпълнение на нишки, организирани в блокове и гридове в до три измерения (x, y, z) [20] [21] [22]. Управлението и достъпът до отделните нишки се осъществява чрез системни променливи като `threadIdx`, `blockIdx`, `blockDim` и `gridDim`, като чрез тях се изчислява глобалният индекс на всяка нишка в грида. Това позволява ефективно разпределение на задачи върху голям брой паралелни изчислителни единици [23] [24].

**Управлението на паметта** в CUDA включва трансфер на данни между хоста (CPU) и устройството (GPU), осъществяван чрез функцията `cudaMemcpy`. Посоката на трансфер може да бъде от хост към устройство, от устройство към хост или между устройства. За управление на паметта се използват функции, аналогични на стандартните в езика C, като `cudaMalloc`, `cudaMemset` и `cudaFree`. Този процес е критичен за производителността на CUDA приложенията, тъй като ефективният трансфер и достъп до паметта значително влияе върху скоростта на изчисленията.

## Архитектура на CUDA

Архитектурата на CUDA се базира на модела SIMT (Single Instruction, Multiple Threads), който разширява традиционните паралелни архитектури като SISD, SIMD, MISD и MIMD. При SIMT една инструкция от *kernel* функция се изпълнява паралелно върху множество нишки, групирани в блокове и гридове. На хардуерно ниво всяка нишка се изпълнява от отделно CUDA ядро, блоковете се обработват от стрийминг мултипроцесори, а цялата грид структура се разгръща върху графичния процесор. Ефективното използване на архитектурните елементи е от съществено значение за постигане на висока производителност[22][25][26].

**Графичният процесор**, например в GeForce RTX 3060, е съставен от множество стрийминг мултипроцесори, всеки от които съдържа компоненти като CUDA ядра, регистри, споделена памет, специализирани единици и уредби за управление на нишките. Наличието на различни типове ядра – като FP32, FP64, Tensor и Int – позволява оптимизация на изчисленията според вида задачи. Всеки мултипроцесор оперира върху локални ресурси, като достъпът до глобалната памет и кеширането чрез L2 Cache играят ключова роля.

**Изчислителните възможности** на дадено устройство се определят от неговите характеристики, достъпни чрез функцията `cudaGetDeviceProperties()`. Сред най-важните параметри са: броят нишки на блок, размерите на блокове и гридове, глобалната и споделената памет, размерът на warp (обикновено 32), както и тактовата честота. Познаването на тези характеристики е важно при преносимостта на CUDA приложения между различни графични устройства и за оптимално използване на наличните ресурси[27].

## Разпределяне на ресурсите и скриване на латентност

Изпълнението на контекста за група от 32 нишки (т.нар. *warp*) включва основни компоненти като програмни броячи, регистри и споделена памет. Всеки warp се обработва от стрийминг мултипроцесор (SM), като неговото състояние се поддържа изцяло в рамките на чипа за цялото времетраене на изпълнението. Това позволява бързо превключване между контексти без значителни загуби на производителност, което е съществено за скриване на латентността при достъп до памет.

Регистровият файл съдържа набор от 32-битови регистри, които се разпределят между нишките. Наред с това, всяка SM разполага с ограничено количество споделена памет, която се разпределя между блоковете от нишки. Възможността за едновременно изпълнение на множество блокове и нишки зависи от броя на регистрите и обема на използваната споделена памет. По този начин, ако се намали използването на регистри на нишка или обемът на споделена памет на блок, това може да позволи изпълнение на повече блокове и нишки паралелно, повишавайки общата изчислителна ефективност.

Максималният брой достъпни регистри и споделена памет е фиксиран и зависи от хардуерните характеристики на графичното устройство. При надвишаване на тези ресурси, дадена *kernel* функция няма да може да се стартира. Информация относно ресурсите на конкретно устройство може да бъде получена чрез функцията `cudaGetDeviceProperties()`, както и от официалната документация на CUDA.

Латентността представлява броя на тактовите цикли между издаването и изпълнението на инструкции. В контекста на CUDA се разграничават два основни типа латентност: аритметична, възникваща при обработка на инструкции от аритметично-логическото устройство (АЛУ), и латентност при достъп до памет, отразяваща времето между заявка за данни и тяхното реално предоставяне. Най-ниска латентност се наблюдава при достъп до споделената памет, което подчертава нейната значимост за оптимизация на изпълнението.

За да се минимизира влиянието на латентността, CUDA използва техника за скриване на латентност чрез превключване между групи нишки (warps), изпълнявани от стрийминг мултипроцесорите (SMs). Когато една група изчаква резултат от дълга аритметична или паметна операция, SM може да превключи към друга готова група от нишки, поддържайки пълна натовареност на изчислителните ресурси. Така се избягва времето на изчакване и се постига висока ефективност.

Определянето на броя нишки и групи от нишки, необходими за скриване на латентността, зависи от конкретните хардуерни характеристики на устройството – брой SM единици, честота на паметта, латентност на DRAM и обем на трансферираните данни. Например, ако аритметичната латентност е 20 такта, при 4 warps на SM са необходими 80 нишки в готовност, което при 13 SM изисква общо 1040 нишки. За латентността на паметта, подобен изчислителен подход се прилага чрез използване на честотната лента и латентността на DRAM, както и типа на използваните данни.

В заключение, оптимизацията на CUDA приложенията следва да се насочи според доминиращите типове латентност – дали са свързани с паметта или с изчисленията – като за всяка категория се прилагат специфични стратегии за скриване на латентността и максимално използване на наличните ресурси.

## Модел на паметта в CUDA

Паметта в CUDA е организирана в йерархична структура, която позволява оптимизация на производителността чрез правилно управление на различните видове памет. Най-бързата памет в тази архитектура са регистрите, които съхраняват локални променливи на нишките. Те са с ограничен брой, като обикновено варират между 63 и 255 на нишка в зависимост от архитектурата. При превишаване на този лимит, излишните променливи се прехвърлят в локалната памет, което води до значително влошаване на производителността. Локалната памет, макар и с подобна логическа функция, физически се намира в DRAM и има висока латентност.

Друг важен компонент е споделената памет, която е с ниска латентност и се намира на чипа. Тя служи като средство за комуникация между нишките в един блок и може да се използва като програмно управляем кеш. Споделената памет е ограничен ресурс – между 48 и 112 KB на стрийминг мултипроцесор, и нейното използване пряко влияе върху броя на блоковете, които могат да се изпълняват едновременно. Освен това, CUDA архитектурата включва и различни типове глобална памет в DRAM – постоянна, текстурна и глобална, всяка с достъп до кешове с цел минимизиране на латентността.

## Локалност и управление на паметта

Концепцията за локалност на данните – пространствена и времева – играе ключова роля в архитектурата на CUDA. Тя се използва за ефективно разпределение на данни в различните слоеве на паметта, като кешове и регистри. Пространствената локалност предполага, че ако една памет е достъпена, то съседните места вероятно също ще бъдат използвани, докато времевата предполага повторен достъп до същото място. Тези принципи обуславят използването на памети с различна латентност и обем – от кешове до DRAM и дори външни устройства като SSD.

В CUDA, паметта се управлява чрез специфични функции. За заделяне на памет в графичното устройство се използва `cudaMalloc()`, а за освобождаване – `cudaFree()`. Трансферът между хоста и устройството се

осъществява с `cudaMemcpy()`. В съвременните устройства честотната лента може да достигне до 484 GB/s, което прави критично важно минимизирането на трансферите между хоста и устройството.

**Разширени техники за управление на паметта**

CUDA поддържа различни методи за оптимизиране на паметта, като фиксирана памет, “Zero copy” и унифицирана памет. Фиксираната памет позволява директен достъп от устройството до хост паметта, като се елиминира нуждата от временни копия. “Zero copy” е разновидност на фиксираната памет, която е картографирана в адресното пространство на устройството, и позволява директна употреба без физически трансфер на данни. Това е полезно при ограничени ресурси, но е неефективно при големи обеми от данни.

Унифицираната памет създава управляема област от паметта, която е достъпна както от CPU, така и от GPU чрез един и същ указател. Тя значително опростява програмирането, като елиминира необходимостта от ръчно копиране между хост и устройство. Паметта може да бъде заделена както статично, така и динамично с `cudaMallocManaged()`, а синхронизацията между хоста и устройството се осъществява автоматично от драйвера.

**Оптимизация и използване на споделената памет**

Споделената памет е от особено значение за оптимизацията на CUDA програмите. Поради ниската си латентност, тя се използва като междинен буфер, кеширащ често използвани данни и намаляващ нуждата от многократен достъп до глобалната памет. Паметта се декларира статично или динамично и е споделена между нишките в рамките на един блок. Нейната ефективност обаче зависи от начина, по който се достъпва – неправилното използване може да доведе до т.нар. "bank conflicts", което намалява производителността. Правилното структуриране и синхронизация при достъп до тази памет е критично за постигането на висока ефективност на изпълнението.

# Глава 3

# Софтуерен инструмент BioPoolSelect

## Биоинформатични анализи и тяхното приложение в биомедицината

Биоинформатичните анализи се превръщат във фундаментален елемент в съвременната биомедицина, като особено значима роля имат при анализа на геномни данни. Тези методи предоставят възможност за детайлно изследване на генетичните особености, свързани с редица заболявания, и намират приложение в различни медицински направления. В сферата на медицинската диагностика, биоинформатичните подходи позволяват откриването на генетични мутации, свързани както с наследствени, така и с онкологични заболявания. Те са неотменна част и от персонализираната медицина, където чрез анализ на индивидуални генетични профили се прогнозира отговорът на пациента към определени лекарствени терапии. Освен това, чрез изследване на генетични вариации, биоинформатиката подпомага идентифицирането на нови терапевтични мишени, което е от ключово значение за разработването на иновативни фармакологични решения.

Интегрирането на биоинформатични алгоритми с високопроизводителни изчислителни платформи значително разширява капацитета за обработка и интерпретация на сложни биомедицински данни. Това улеснява дълбокото разбиране на молекулярните механизми, лежащи в основата на заболяванията, като по този начин подпомага разработването на таргетирани и ефективни лечебни стратегии. Съвременните изчислителни технологии, включително машинното обучение и облачните решения, предоставят необходимата инфраструктура за реализация на тези анализи в реално време и в мащаб, подходящ за клинични и научни цели.

## Разработване на софтуерен инструмент за анализ на геномни варианти

Като част от настоящото изследване е разработен специализиран софтуерен инструмент, предназначен за филтриране и анализ на геномни варианти, съобразени с конкретни изследователски нужди. Основните функционалности на инструмента включват автоматизирано филтриране на мутации, които са асоциирани с редица заболявания, както и идентификация на патогенни фактори на базата на генетични вариации – например при пациенти с невродегенеративни заболявания като болестта на Алцхаймер. Освен това,

софтуерът предлага интерактивно сортиране и визуализация на резултатите, улеснявайки процеса на генетична интерпретация.

Разработената платформа използва оптимизирани алгоритми за обработка на големи обеми геномни данни, като същевременно интегрира машинно-обучителни модели и поддръжка на облачна инфраструктура. Това осигурява не само висока ефективност при анализа на секвенционни данни, но и възможност за мащабируемост в зависимост от нуждите на конкретното изследване или клинична практика. Софтуерният инструмент значително подобрява възможностите за биоинформатичен анализ, като съчетава скорост, прецизност и адаптивност. По този начин той допринася за по-доброто разбиране на генетичната етиология на заболяванията и подпомага разработването на персонализирани терапевтични подходи.

**Молекулярни механизми и регулаторни процеси в биоинформатиката**

В основата на биоинформатиката стои задълбоченото разбиране на молекулярните механизми, които определят взаимодействието между генетичната информация и клетъчните функции. Дезоксирибонуклеиновата киселина (ДНК), като носител на наследствената информация, участва активно в регулацията на фенотипната изява чрез сложна мрежа от регулаторни процеси. Биоинформатичните методи осигуряват мощни инструменти за анализ на тези процеси, позволявайки изчислително моделиране на ключови етапи от функционирането на клетката.

Централната догма на молекулярната биология [73] описва последователността от процеси, чрез които генетичната информация се реплицира, транскрибира и транслативно интерпретира. Тази концептуална рамка е основополагаща и за биоинформатиката, където чрез разработване на алгоритми и модели се анализира поведението на нуклеинови и белтъчни последователности. Приложението на скрити Маркови модели, невронни мрежи и други машинно-обучителни подходи позволява автоматизирано предсказване на гени, функционални домейни и протеинови структури, което е от особено значение за молекулярната диагностика и биомедицинските изследвания.

Допълнително, транслационният процес и регулацията на транскрипцията са от особено значение за разбирането на клетъчната експресия. Гените в еукариотните организми са организирани в структурни зони – 5′-UTR, кодиращ регион и 3′-UTR – всяка от които играе специфична роля в

регулацията и стабилността на информационната РНК. Промоторните региони, свързани с началото на транскрипцията, също представляват обект на интензивен биоинформатичен анализ, насочен към идентифициране на регулаторни елементи и транскрипционни фактори [74].

Сложността на геномните структури, особено при висши еукариоти, налага използването на хибридни подходи за идентификация и анотация на гени. Сред тях са алгоритми за хомологично сравнение, детекция на регулаторни мотиви и анализ на сплайсинг сигнали. Макар съществуващите методи да предлагат висока чувствителност, все още няма напълно надежден алгоритъм за точна и автоматизирана идентификация на всички функционални елементи в генома. Ето защо биоинформатиката остава динамично развиваща се област, в която се изследват и прилагат нови изчислителни стратегии за по-прецизен анализ на генетичните данни и разбиране на молекулярните основи на биологичните процеси [75].

Молекулярните механизми и регулаторните процеси в биоинформатиката представляват основа за изследване на взаимодействието между генетичната информация и клетъчните функции, като чрез изчислителни методи и алгоритми се моделират процеси като репликация, транскрипция, транслация и генна регулация, с цел по-добро разбиране и предсказване на биологични явления и заболявания.

## Софтуерен инструмент BioPoolSelect

Софтуерният инструмент **BioPoolSelect** е създаден като отговор на необходимостта от ефективна обработка на големи геномни масиви, които традиционните методи и програми не могат да анализират поради техния обем и комплексност. Разработката му е базирана на програмния език *Python*, поради неговата висока гъвкавост и богат набор от библиотеки, подходящи за научни изчисления, статистически анализ и визуализация на данни. Библиотеки като *NumPy*, *Pandas*, *Matplotlib* и *Scikit-learn* осигуряват необходимата функционалност за числени изчисления, извличане на структурирана информация, конструиране на предсказващи модели и представяне на резултатите в удобен за интерпретация формат.

BioPoolSelect предлага иновативен подход за автоматизирано филтриране и извличане на ключова информация от геномни файлове, като се фокусира върху идентифицирането на патогенни варианти, свързани с невродегенеративни и други заболявания. При разработването му са

използвани данни от AADR (Alien Ancient DNA Resource) [76], като софтуерът позволява локализиране на генетични варианти и анализ на тяхното разположение спрямо кодиращите области на гените, както и ефектите върху белтъчната експресия. Допълнително се извършва анализ на популационната честота на вариантите на базата на данни от 1000 Genomes Project и ClinVar [77], като по този начин се осигурява контекстуална оценка на вероятната патогенност.

Сред основните функционалности на инструмента са възможността за избор на специфични геномни характеристики, филтриране на информацията и създаване на нови, по-компактни файлове, които могат да бъдат обработвани с широкоразпространени инструменти. Това включва и опция за разделяне на големи файлове на по-малки части с цел по-лесна манипулация и сравнение между различни геномни проби. Програмата дава възможност на потребителя да избира кои характеристики да бъдат включени в анализа – например, местоположение на варианта в генома, промени в нуклеотидната последователност, честоти по популации и референтни стойности – и автоматично генерира файл, съдържащ само релевантната информация.

Софтуерът *BioPoolSelect* представлява значим принос в областта на биоинформатиката, като съчетава съвременни програмни практики с конкретни нужди в анализа на древна и съвременна ДНК. Неговата гъвкавост, мащабируемост и висока производителност го правят подходящ както за научни изследвания, така и за клинична генетика и персонализирана медицина.

Софтуерът **BioPoolSelect** е използван за селектиране на редки геномни варианти, свързани със заболявания, които имат ниска честота в популацията и вероятно притежават вредно въздействие върху човешкия организъм. В изследването е използвана базата данни *DisGeNET [78]* – мащабна платформа за асоциации между гени и човешки заболявания – с цел идентификация на генетични варианти, еднозначно свързани с невродегенеративни състояния като болестта на Алцхаймер, Паркинсон и епилепсия. Получените резултати са сравнени с геномите на неандерталци, за да се провери дали техният генетичен материал съдържа варианти, патогенни за съвременните хора и асоциирани с моногенни заболявания. Такива сравнителни геномни анализи разкриват важни аспекти от еволюцията и етиологията на генетичните предразположения към дегенеративни заболявания [79].

**Резултати от изследването**

Анализът на обогатени с единични нуклеотидни полиморфизми (SNP) геномни данни от архаични хоминини разкри наличието на патогенни мутации, асоциирани със заболявания при съвременните хора. Установени са варианти в пет гена с потенциално значими клинични последствия, като при архаичните проби се наблюдава необичайно висока честота на тези мутации спрямо съвременните човешки популации. По-конкретно, девет проби носят вариант в гена *F5*, асоцииран с повишен риск от венозен тромбоемболизъм (ВТЕ), а пет – вариант в гена *GALT*, свързан с галактоземия на Duarte.

Резултатите подчертават значително по-високата честота на определени патогенни варианти в архаичните геноми, като шест от десет изследвани проби съдържат множество такива мутации – явление, което е рядко наблюдавано в съвременни популации. Възможно обяснение за това натрупване са по-леките клинични прояви в комбинация с ограничен размер на популациите и повишен инбридинг. Освен това, данните повдигат хипотезата, че някои от тези варианти може да са резултат от интрогресия от други архаични човешки видове, което предполага потенциална роля на архаичния генетичен принос за съвременната заболеваемост.

Интегрирането на древни геномни данни с клинични, епидемиологични и популационногенетични подходи разширява разбирането за еволюционната история на рода *Homo*. Това дава възможност да се проследят взаимодействията между генетичния състав, културната еволюция и факторите на средата, които оформят фенотипните вариации в контекста на здравето и болестите.

# Глава 4

# Алгоритъм Needleman-Wunsch и техники за разпаралеляване и оптимизация

## Същност на алгоритъма Needleman-Wunsch

Алгоритъмът Needleman-Wunsch представлява класически подход за глобално подравняване на биологични секвенции [17][80], основан на динамично програмиране. Той се използва, когато се изисква оптимално съвпадение по цялата дължина на две секвенции, като оценява всички възможни подравнявания и избира това с най-високата оценка. В контекста на компютърните науки, подравняваните секвенции се третират като низове със зададени дължини, например S[1…n] и T[1…m], като се изгражда матрица V(i,j), в която се съхраняват оптималните стойности за частични подравнявания.

Изчисленията в матрицата се извършват рекурсивно, като се разглеждат два основни сценария: подравняване с празен низ (крайни случаи) и общ случай, при който се взима максималната стойност измежду три възможности – съвпадение или несъвпадение, вмъкване и изтриване. За всяка позиция в матрицата се използват предварително дефинирани стойности: MATCH за съвпадение, MISMATCH за несъвпадение и GAP за пропуск, които определят оценката на съответната операция. Например, ако MATCH = 2, MISMATCH = –3 и GAP = –2, стойността V(1,1) се изчислява като максимум от трите съответстващи стойности – резултатът е 2 при съвпадение.

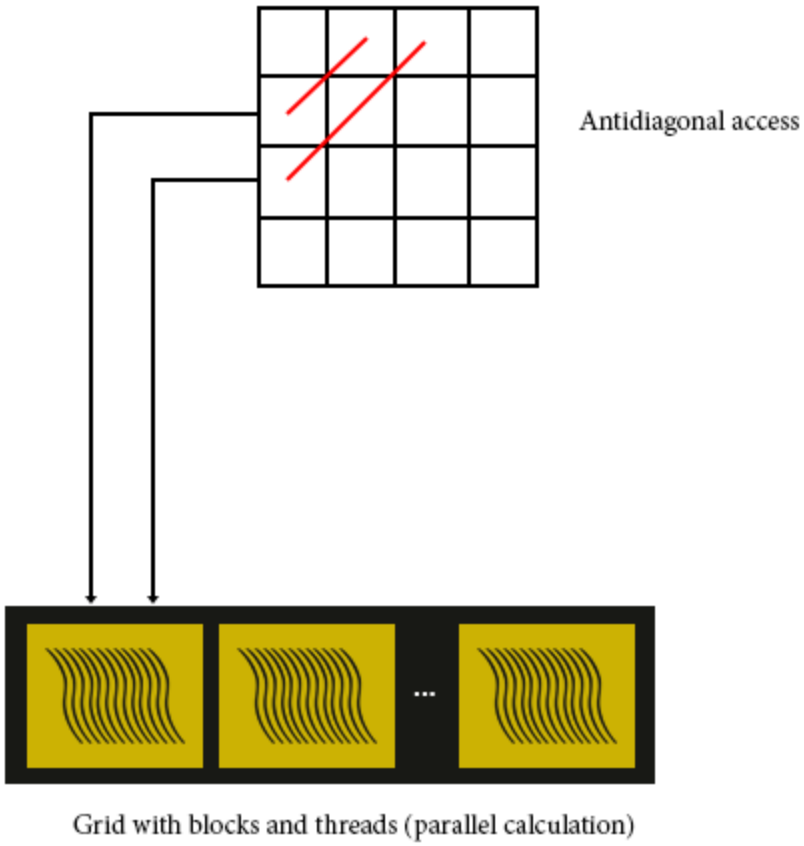
Крайният резултат от оптималното глобално подравняване е записан в долния десен ъгъл на матрицата – V(n,m). За да се възстанови оптималното подравняване, се проследява пътят обратно към началото чрез система от стрелки: диагонални при съвпадение/несъвпадение, хоризонтални при вмъкване и вертикални при изтриване [81]. Този метод осигурява точно и ефективно сравнение на биологични последователности, което го прави основополагащ в анализа на генетични данни и еволюционни връзки.

**Разпаралеляване на алгоритъма Needleman-Wunsch върху графичен ускорител NVIDIA с използване на CUDA C**

Архитектурата на CUDA, разработена от NVIDIA, е специално проектирана за мащабна паралелна обработка, като позволява едновременното изпълнение на хиляди нишки върху графичния процесор (GPU). Тази архитектура предлага значителни предимства при изчислително интензивни задачи, но поставя предизвикателства при реализацията на алгоритми за динамично програмиране като Needleman-Wunsch, поради наличието на силни зависимости между елементите в изчислителната матрица. Всяка стойност в матрицата зависи от вече изчислени съседни клетки, което ограничава директното използване на паралелна обработка.

За да се реализира ефективна паралелизация, е необходимо алгоритъмът да бъде трансформиран така, че да разделя изчисленията на независими етапи, съобразени със зависимостите в данните. Един от утвърдените методи за това е т.нар. антидиагонален (wavefront) подход. Чрез него клетките в матрицата се обработват по антидиагонали, т.е. всички елементи с равна сума от индексите си (i+j=const) се изчисляват паралелно, тъй като не зависят един от друг. Това позволява ефективно натоварване на CUDA нишките, минимизиране на необходимостта от синхронизация и максимално използване на наличните изчислителни единици на GPU [82].

Използването на CUDA C в този контекст предоставя директен контрол върху управлението на нишките, блоковете и паметта, което дава възможност за фина настройка на производителността. Чрез внимателно управление на изчислителната последователност и синхронизацията между нишките се постига значително ускорение на алгоритъма в сравнение с традиционната му CPU реализация. Този подход прави възможно прилагането на Needleman-Wunsch върху големи биоинформатични данни, като същевременно съкращава времето за изчисления и подобрява мащабируемостта на анализа.




Фиг. 1. Паралелизация в няколко етапа – антидиагонален достъп до елементите на матрицата на Needleman-Wunsch и препращането им в паралелната архитектура на CUDA

## Главна функция и паралелни етапи на изчисление

В главната функция се реализира основната логика на алгоритъма за паралелно подравняване на биологични последователности чрез използване на графичен процесор (GPU). В началото се извършва прочитане на два FASTA файла, съдържащи съответните последователности, които ще бъдат подравнени. На тази база се изчисляват размерите на матрицата за динамично програмиране, необходими за прилагане на алгоритъма на Нидълман-Вунш (Needleman-Wunsch).

С оглед осигуряване на ефективна обработка в среда с графично ускорение, се заделя динамична памет, както в хост паметта, така и в паметта на графичното устройство. След инициализацията, данните се трансферират към GPU и се стартира първият паралелен етап на изчисление. За организацията на последващите паралелни изчисления се използва т.нар. антидиагонален

достъп, при който всяка антидиагонала се обработва като едномерен масив. Тази организация способства за повишаване на ефективността на векторизираното изчисление, като се избягва необходимостта от последователен достъп до съседни елементи.

**Паралелен етап 1: Инициализация на матрицата**

В този етап се използват две функции:

1. `cudaError_t NWCudaInitial(int *matrixNW, unsigned int aSize, unsigned int bSize)`

Функцията има за цел да задели линейна памет в графичното устройство за цялата матрица. Размерите ѝ се определят на база дължините на двете последователности. След успешно заделяне на паметта, се извиква kernel-функцията `fillMatrix`, която инициализира първия ред и първата колона на матрицата с фиксираната стойност на параметъра GAP. След нейното изпълнение се използва `cudaDeviceSynchronize()`, за да се осигури, че всички операции върху устройството са завършени преди да продължи изпълнението на процесора.

2. `__global__ void fillMatrix(int *matrixNW, int aNWsize, int bNWsize)`

Kernel-функцията запълва паралелно първия ред и първата колона на матрицата. Тъй като тези стойности зависят единствено от параметъра GAP и не изискват допълнителни зависимости, те са подходящи за паралелизация.

**Паралелен етап 2: Обработка на първата половина от матрицата**

В този етап се прилага антидиагонален достъп до горната част на матрицата. Главната функция използва двоен цикъл, който извършва векторизация на всяка антидиагонала. След всяка итерация се извиква функцията `NWCudaAntidiagonal`, която подготвя съответния антидиагонал за изчисление, и `cudaDeviceSynchronize()`, която гарантира завършване на GPU операциите преди преминаване към следващата антидиагонала.

*Използвани функции:*

1. `cudaError_t NWCudaAntidiagonal(...)`

Функцията заделя памет за съответните позиции по x и y и стойностите на текущата антидиагонала. Данните се подготвят като едномерни масиви, които се предават на GPU. Функцията извиква kernel-функцията `filldiagMatrix`, която извършва паралелното изчисление на стойностите.

2. `__global__ void filldiagMatrix(...)`

Тази kernel-функция изчислява стойностите на антидиагоналата, като използва съседните клетки от матрицата съгласно принципа на динамично програмиране. Паралелното изпълнение се базира на броя елементи в съответната антидиагонала, като всяка нишка отговаря за конкретен елемент.

### Паралелен етап 3: Обработка на втората половина от матрицата

Този етап се реализира аналогично на предходния, с разликата, че антидиагоналите се генерират от долната част на матрицата към горната. Главната функция използва отново двоен цикъл, като във всяка итерация се извикват `NWCudaAntidiagonal2` и `filldiagMatrix2`.

*Използвани функции:*

1. `cudaError_t NWCudaAntidiagonal2(...)`

Аналогична на `NWCudaAntidiagonal`, но адаптирана за втората половина на матрицата.

2. `__global__ void filldiagMatrix2(...)`

Функция, идентична по структура на `filldiagMatrix`, адаптирана за изчисление на долните антидиагонали на матрицата.

### Функции за оптимизация на грида

За да се осигури оптимална конфигурация на блоковете и нишките при изпълнение на паралелните функции, са реализирани следните помощни функции:

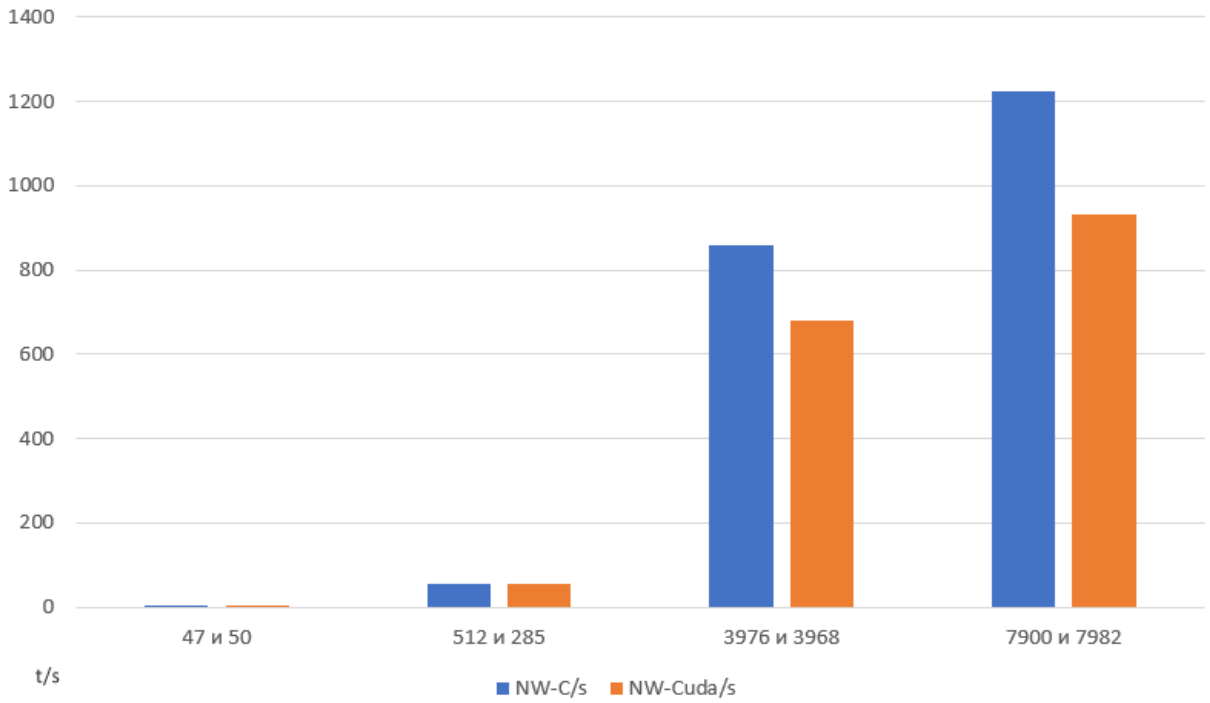
1. `dim3 GridSize(int aSize, int bSize)` – Изчислява подходяща размерност на грида (едномерна, двумерна или триизмерна), в зависимост от размерите на матрицата и ограниченията на CUDA.
2. `bool exceedsDimx(int aSize, int bSize)` – Проверява дали броят на нишките по оста x надвишава допустимия праг.
3. `int calcDimx(int aSize, int bSize)` – Изчислява необходимия брой нишки по оста x на база размерите на матрицата.
4. `int calcNumBlocks(int aSize, int bSize)` – Определя оптималния брой блокове, при препоръчителен размер на блок от 512 нишки, което съответства на архитектурната организация на CUDA.
5. `bool exceedsDimyz(int a)` – Проверява дали размерите по оста y или z надвишават допустимите стойности (65535).

**Резултати.**

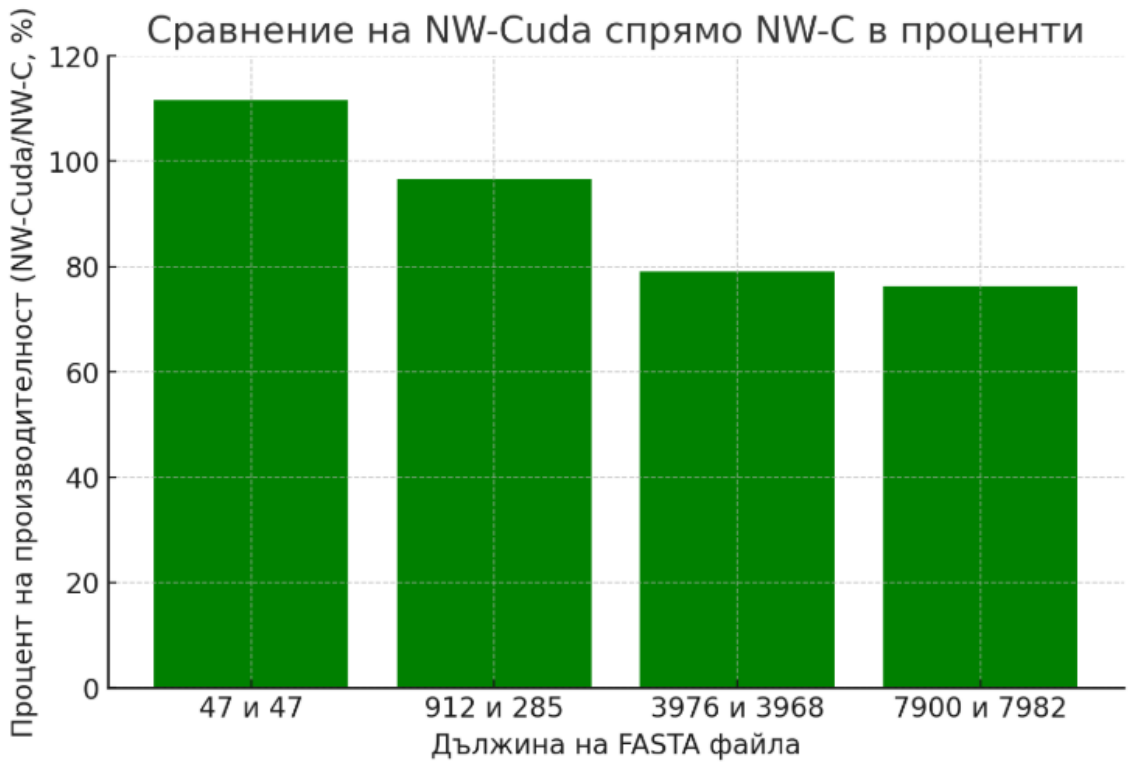
След проведени тестове се получиха следните сравнителни резултати, като сравнението е извършено спрямо алгоритъм на Needleman-Wunsch, написан на C.

| Дължина на FASTA файла | NW-C/sec | NW-Cuda/sec |
|---|---|---|
| 47 и 47 | 3.03 | 3.38 |
| 912 и 285 | 57.03 | 55.14 |
| 3976 и 3968 | 860.02 | 680.05 |
| 7900 и 7982 | 1224.37 | 933.20 |

Табл. 1. Сравнителни резултати на времето за изпълнение между алгоритъм NW, реализиран на C и алгоритъм NW, реализиран на CUDA C.




Фиг. 2. Диаграма към таблица 1, показваща времето за изпълнение на двете версии.




Фиг. 3. Диаграма към таблица 1, показваща времето за изпълнение на двете версии в проценти.

Видно е, че паралелният алгоритъм е значително по-бърз за всички сравнени последователности.

## Използване на изкуствен интелект и алгоритъм Needleman-Wunsch

Паралелната реализация на алгоритми върху графични процесори (GPU) налага прилагането на внимателно структурирани изчислителни стратегии, с цел избягване на състезателни условия (race conditions), характерни за архитектурата на този тип хардуер. В този контекст, антидиагоналният (Wavefront) подход се утвърждава като ефективна техника за управление на зависимостите в изчислителния процес. За разлика от традиционния редово-ориентиран обход, който предизвиква конкуренция за достъп до споделени клетки в матрица, антидиагоналната организация осигурява независимост между елементите в рамките на всяка антидиагонал. Това позволява равномерно разпределение на задачите между нишките и предотвратява конфликти при четене и запис на данни.

Същността на алгоритъма включва изчисление на стойности в двумерна матрица въз основа на стойностите на три съседни клетки – горна, лява и горна ляво-диагонална. Прилагането на Wavefront стратегията гарантира, че към момента на обработка на дадена антидиагонал всички необходими данни от предходните антидиагонали вече са налични. Това позволява всяка антидиагонала да се обработва паралелно, като се използва пълноценно наличният хардуерен паралелизъм. Особено важно е, че при достигане до централните антидиагонали на матрицата се активират повече CUDA ядра, поради увеличената дължина на тези антидиагонали, което води до значително повишаване на изчислителната ефективност.

За да се постигне оптимална производителност на GPU-базирания алгоритъм, е необходимо неговото профилиране с помощта на специализирани инструменти. Решенията на Nvidia – Nsight Systems и Nsight Compute – предоставят широк набор от възможности за анализ на производителността на различни нива. Те дават достъп до стотици хардуерно-специфични метрики, визуализации на изчислителните процеси и топологични зависимости, като така подпомагат както макро-, така и микрооптимизацията на алгоритмите. Този подход е критично важен дори когато първоначалните резултати от изпълнението съответстват на очакванията, тъй като позволява откриване на скрити тесни места и неефективности.

След приключване на профилирането, алгоритъмът се мащабира за обработка на голям брой биоинформатични задачи, по-специално изравняване на биологични последователности с дължини между 9000 и 10 000 символа. Получените резултати се интегрират в изчислителен конвейер, който включва и модули за сравнение със стойности на сходство, генерирани от модели на изкуствен интелект. На този етап се дефинира индекс на изравняване на последователности (Sequence Alignment Index – SAI), изчисляван като разлика между коефициентите на сходство преди и след изравняването. Анализът на корелацията между този индекс и резултатите от изравняването цели извеждане на значими зависимости в данните. Крайната цел е изграждането на прогностичен модел за стойността на SAI, базиран единствено на резултатите от изравняването, което би допринесло за по-добро разбиране на взаимовръзките между структурното сходство и качеството на подравняване в биологични данни [83-85].

## Същност на Llama2

Llama 2 представлява съвременна реализация на големи езикови модели (LLM), разработена от Meta AI с цел разширяване на достъпа до мощни инструменти за обработка на естествен език. Публикуван през 2023 г., този модел се отличава с отворена политика на лицензиране, като позволява както научна, така и търговска употреба. В основата си, Llama 2 е семейство от предварително обучени и фино настроени трансформерни модели, способни да изпълняват разнообразни NLP задачи — от текстово генериране и довършване на изречения до кодиране и семантичен анализ. Разработката му цели не само технологичен напредък, но и демократизация на генеративния изкуствен интелект чрез достъпен и ефективен хардуерен отпечатък.

Llama 2 включва както базови, така и фино настроени чат модели, които наследяват и усъвършенстват архитектурните принципи на предшественика си — LLaMa 1. За разлика от водещите патентовани модели като GPT, Claude или Bard, които са недостъпни за свободна разработка и анализ, Llama 2 предоставя публичен достъп до моделните тегла и архитектура, като по този начин насърчава иновации, прозрачност и колективен напредък в изследванията. Моделите се предлагат в три мащабируеми конфигурации — 7B, 13B и 70B параметри, което улеснява тяхното използване от малки организации и академични среди, дори при ограничени ресурси.

Технически, Llama 2 е изграден като авторегресивен причинно-следствен езиков модел, използващ трансформерна архитектура. Чрез процеса на

самоконтролирано предварително обучение върху 2 трилиона токена от публично достъпни източници, моделът усвоява езикови закономерности и семантична структура. Базовата му функционалност се изразява в автодовършване на текст, без специфична насоченост към изпълнение на задачи. За да бъде адаптиран към практически приложения — като диалог, кодиране или следване на инструкции — се използват подходи за фино настройване, сред които обучение с наблюдение (SFT) и обучение с подсилване от човешка обратна връзка (RLHF).

Llama 2 вече е основа за редица водещи LLM с отворен код, включително Alpaca, Vicuna, Orca и WizardLM. Тези производни модели демонстрират висока ефективност при минимални разходи за обучение и често се конкурират с търговски решения от най-висок клас. Те утвърждават Llama 2 като гъвкава, икономична и продуктивна основа за разработка на специализирани езикови системи, които обслужват широк спектър от изследователски и приложни нужди.

# Заключение

За ефективна паралелна реализация на алгоритъма на Needleman-Wunsch [89] върху графичен процесор (GPU) се налага прилагането на антидиагонален (Wavefront) подход, поради присъщите на алгоритъма зависимости между матричните елементи, които създават предпоставки за състезателни условия при едновременен достъп. Алгоритъмът изисква за изчисляването на стойността на дадена клетка в матрицата наличието на резултатите от три съседни клетки – горна, лява и горна ляво-диагонална. Тези зависимости възпрепятстват прилагането на класически редово или колонно ориентирани стратегии за обход, тъй като те не позволяват едновременна обработка без конфликти при четене и запис на данни.

Антидиагоналният подход предлага ефективно решение, при което елементите от един и същи антидиагонал могат да бъдат изчислявани паралелно, тъй като всички нужни зависимости вече са удовлетворени от предходния антидиагонал. Това води до естествена възможност за разпределение на задачите между паралелни изчислителни нишки, което е в пълно съответствие с архитектурата на графичните процесори.

Особено предимство на този подход е способността му да мащабира изчислителния процес съобразно големината на входните последователности. С нарастването на дължината на антидиагоналите към централната част на матрицата се увеличава и броят на независимите задачи, което позволява използване на все по-голяма част от наличните CUDA ядра. Това води до висока степен на паралелизъм и оптимално използване на изчислителния капацитет на GPU. Тъй като графичните процесори са специализирани за изпълнение на голям брой операции с плаваща запетая, антидиагоналният подход се явява оптимална стратегия за постигане на максимална производителност при паралелна реализация на алгоритъма Needleman-Wunsch.

# Приноси

**Научно-приложни приноси:**

- **Създадена е оптимизация на паралелна версия на алгоритъма на Needleman-Wunsch** за графични процесори (GPU). Въвеждане и обосновка на използването на антидиагоналния подход (Wavefront) за ефективно паралелизиране на изчисленията. Този подход осигурява по-добро използване на ресурсите на GPU, като елиминира състезателните условия и максимизира използването на CUDA ядра.
- **Въведен е индекс на изравняване на последователности**. Дефинира нова метрика, която измерва разликата между коефициентите на сходство преди и след подравняването, което може да помогне за по-доброто разбиране на връзката между сходството на последователностите и ефективността на подравняването.
- **Разработено е ръководство по дисциплината "Програмиране на съвременни хетерогенни архитектури",** което запознава студентите с архитектурните характеристики на хетерогенните изчислителни системи, модели за паралелно програмиране върху различни архитектурни компоненти, техники за разпределяне на натоварването и ефективно управление на памет и ресурси.

**Приложни приноси**:

- **Разработен е софтуерен инструмент за геномен анализ (BioPoolSelect).** Създаден е нов софтуер, който позволява селектиране на специфични геномни характеристики за анализ, като намалява размера на първичните файлове и улеснява обработката на данните. Инструментът дава възможност за избор на конкретни характеристики, свързани с геномни варианти, като тяхната позиция, кодираща област, популационна честота и наличност в клинични бази данни. Инструментът дава възможност да се използва като помощно средство при подравняване на последователности с алгоритъм Needleman-Wunsch в случай на големи и невъзможни за обработка файлове чрез разбиването им на по-малки.

- **Представен е сравнителен анализ на геноми.** Софтуерът BioPoolSelect автоматизира процеса на обработка на големи геномни файлове, като прави възможно по-бързо и ефективно сравнение на геномни данни.

# Списък на отпечатани и приети за печат научни трудове

## А. Научни публикации:

[p1] Kolev, V., Marinova, M., & Pardo, E. (2023, December). Optimal global sequence alignments of COVID-19 nucleotide sequences using the Needleman-Wunsch algorithm and program parallelization. In AIP Conference Proceedings, AMEE 2022 (Vol. 2939, No. 1). AIP Publishing. doi.org/10.1063/5.0178887

[p2] Pardo, E., Marinova, M. (2024, May). DNA Segment Search Program. In AIP Conference Proceedings (Vol. 3274, Issue 1, No 040005), ISSN: 15517616 0094243X, DOI: 10.1063/5.0258840

[p3] Pardo, E. (2024, May) Overview of the Modern Approach of Sequence Alignment Algorithms. In AIP Conference Proceedings (Vol. 3274, Issue 1, No 040006), ISSN: 15517616 0094243X, DOI: 10.1063/5.0258840

[p4] Preprint Pardo, E., Milev, V., Kolev, V., Marinova, M. Analyzing at Scale the Effects of Optimal Global Sequence Alignment on Sequence Similarity using a GPU Optimized Implementation of the Needleman-Wunsch Algorithm and the SBERT module Preprint – TechSys 2025

## Б. Публикувано университетско учебно пособие

[p5] Маринова, М., Пардо, Е. 2023 България, София. Ръководство за лабораторни упражнения по програмиране на съвременни хетерогенни архитектури. Първо издание. ISBN 978-619-167-526-5. https://e-university.tu-sofia.bg/e-publ/search-det.php?id=12109

# Annotation

This dissertation focuses on the design and implementation of a high-performance parallel approach for the Needleman–Wunsch algorithm on graphics processing units (GPUs), utilizing the CUDA architecture. The primary objective of the research is to enhance computational efficiency in global sequence alignment by developing an optimized parallel implementation of this classical algorithm, widely applied in the field of bioinformatics.

The first chapter presents the theoretical foundation of the study through a comprehensive literature review covering the structure and informational function of nucleic acids, sequencing techniques, molecular evolution, and the central dogma of molecular biology. Special emphasis is placed on the application of CUDA technology in bioinformatics and the role of alignment algorithms.

The second chapter explores the CUDA architecture and principles of parallel programming, including resource and memory management, latency hiding, and the use of shared memory—all of which are essential for the efficient execution of computationally intensive algorithms.

The third chapter is devoted to the design and development of the software tool BioPoolSelect, which integrates parallel algorithms for the analysis of genomic variants, supporting research in molecular biology and biomedicine.

The fourth chapter provides an in-depth description of the developed parallel implementation of the Needleman–Wunsch algorithm. It details the three main parallel computation stages, optimization strategies for grid configuration, and the use of shared memory. The chapter also includes an analysis of experimental results obtained from real biological datasets. Furthermore, the potential integration of advanced language models, such as Llama2, is examined to enrich the analytical capabilities of the platform.

The final section of the dissertation outlines the conclusions and summarizes the scientific and applied contributions in the areas of parallel programming, sequence alignment algorithms, and the development of specialized software solutions for bioinformatics.

# Използвани източници:


[1] A. Travers and G. Muskhelishvili, “DNA structure and function,” *The FEBS Journal*, vol. 282, no. 12, pp. 2279–2295, 2015, doi: https://doi.org/10.1111/febs.13307.

[2] J. D. WATSON and F. H. C. CRICK, “Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid,” *Nature*, vol. 171, no. 4356, pp. 737–738, Apr. 1953, doi: 10.1038/171737a0.

[3] H. Kamminga, S. de Chadarevian, and G. Whipple Museum of the History of Science (Cambridge, *Representations of the Double Helix*. Whipple Museum of the History of Science, 2002. [Online]. Available: https://books.google.bg/books?id=c1lqNAAACAAJ

[4] J. Gorodkin and W. L. Ruzzo, *RNA sequence, structure, and function: computational and bioinformatic methods*. Springer, 2014.

[5] D. W. Mount, “Bioinformatics-sequence and genome analysis .,” 2004.

[6] W. Haque, A. Aravind, and B. Reddy, “Pairwise sequence alignment algorithms: a survey,” in *Proceedings of the 2009 Conference on Information Science, Technology and Applications*, in ISTA ’09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 96–103. doi: 10.1145/1551950.1551980.

[7] S. Henikoff and J. G. Henikoff, “Amino acid substitution matrices from protein blocks.,” *Proceedings of the National Academy of Sciences*, vol. 89, no. 22, pp. 10915–10919, 1992.

[8] M. Fakirah, M. A. Shehab, Y. Jararweh, and M. Al-Ayyoub, “Accelerating Needleman-Wunsch global alignment algorithm with GPUs,” *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–5, 2015.

[9] T. R. P. Siriwardena and D. N. Ranasinghe, “Accelerating global sequence alignment using CUDA compatible multi-core GPU,” *2010 Fifth International Conference on Information and Automation for Sustainability*, pp. 201–206, 2010.

[10] B. Strengholt and M. Brobbel, “Acceleration of the Smith-Waterman algorithm for DNA sequence alignment using an FPGA platform,” *Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science*, 2013.

[11] E. F. de O. Sandes, G. Miranda, X. Martorell, E. Ayguadé, G. Teodoro, and A. C. M. A. de Melo, “CUDAlign 4.0: Incremental Speculative Traceback for Exact Chromosome-Wide Alignment in GPU Clusters,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 2838–2850, 2016.

[12] W. R. Pearson, “Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms,” *Genomics*, vol. 11, no. 3, pp. 635–650, 1991, doi: https://doi.org/10.1016/0888-7543(91)90071-L.

[13] S. El-Metwally, O. M. Ouda, and M. Helmy, *Next generation sequencing technologies and challenges in sequence assembly*, vol. 7. Springer Science & Business, 2014.

[14] W. S. Martins, J. B. del Cuvillo, F. J. Useche, K. B. Theobald, and G. R. Gao, “A multithreaded parallel implementation of a dynamic programming algorithm for sequence comparison,” in *Biocomputing 2001*, World Scientific, 2000, pp. 311–322.

[15] B. Chen, Y. Xu, J. Yang, and H. Jiang, “A new parallel method of smith-waterman algorithm on a heterogeneous platform,” presented at the Algorithms and Architectures for Parallel Processing: 10th International Conference, ICA3PP 2010, Busan, Korea, May 21-23, 2010. Proceedings. Part I 10, Springer, 2010, pp. 79–90.

[16] Y. Jararweh, M. Al-Ayyoub, M. Fakirah, L. Alawneh, and B. B. Gupta, “Improving the performance of the needleman-wunsch algorithm using parallelization and vectorization techniques,” *Multimedia Tools and Applications*, vol. 78, pp. 3961–3977, 2019.

[17] V. Gancheva and I. Georgiev, “Multithreaded parallel sequence alignment based on needleman-wunsch algorithm,” presented at the 2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE), IEEE, 2019, pp. 165–169.
[18] S. F. Altschul *et al.*, “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs,” *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, Sep. 1997, doi: 10.1093/nar/25.17.3389.
[19] E. S. Donkor, N. T. Dayie, and T. K. Adiku, “Bioinformatics with basic local alignment search tool (BLAST) and fast alignment (FASTA),” 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:55990007
[20] H. Khaled, R. El Gohary, N. Badr, and H. Faheem, “Accelerating pairwise DNA Sequence Alignment using the CUDA compatible GPU,” *International journal of computer applications*, vol. 84, no. 1, 2013.
[21] C. Ling, K. Benkrid, and T. Hamada, “A parameterisable and scalable Smith-Waterman algorithm implementation on CUDA-compatible GPUs,” presented at the 2009 IEEE 7th Symposium on Application Specific Processors, IEEE, 2009, pp. 94–100.
[22] N. Ahmed, J. Lévy, S. Ren, H. Mushtaq, K. Bertels, and Z. Al-Ars, “GASAL2: a GPU accelerated sequence alignment library for high-throughput NGS data,” *BMC Bioinformatics*, vol. 20, no. 1, p. 520, Oct. 2019, doi: 10.1186/s12859-019-3086-9.
[23] M. Nemirovsky and D. Tullsen, *Multithreading architecture*. Springer Nature, 2022.
[24] S. N. Devi and S. Rajagopalan, “A modified dynamic parallel algorithm for sequence alignment in biosequences,” *International Journal of Computer Applications*, vol. 60, no. 18, 2012.
[25] V. W. Lee *et al.*, “Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU,” *Proceedings of the 37th annual international symposium on Computer architecture*, 2010, [Online]. Available: https://api.semanticscholar.org/CorpusID:10039635
[26] S. Wang, A. Prakash, and T. Mitra, “Software Support for Heterogeneous Computing,” *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 756–762, 2018.
[27] C. Gregg and K. M. Hazelwood, “Where is the data? Why you cannot debate CPU vs. GPU performance without the answer,” *(IEEE ISPASS) IEEE INTERNATIONAL SYMPOSIUM ON PERFORMANCE ANALYSIS OF SYSTEMS AND SOFTWARE*, pp. 134–144, 2011.
[28] C. W. Yu, K. H. Kwong, K. H. Lee, and P. H. W. Leong, “A Smith-Waterman Systolic Cell,” in *New Algorithms, Architectures and Applications for Reconfigurable Computing*, P. Lysaght and W. Rosenstiel, Eds., Boston, MA: Springer US, 2005, pp. 291–300. doi: 10.1007/1-4020-3128-9_23.
[29] Y. Liu and B. Schmidt, “GSWABE: faster GPU-accelerated sequence alignment with optimal alignment retrieval for short DNA sequences,” *Concurrency and Computation: Practice and Experience*, vol. 27, no. 4, pp. 958–972, 2015.
[30] A. Dhraief, R. Issaoui, and A. Belghith, “Parallel computing the longest common subsequence (LCS) on GPUs: efficiency and language suitability,” presented at the The 1st International Conference on Advanced Communications and Computation (INFOCOMP), 2011, pp. 143–148.
[31] Y. Li, J. M. Patel, and A. Terrell, “Wham: a high-throughput sequence alignment method,” *ACM Transactions on Database Systems (TODS)*, vol. 37, no. 4, pp. 1–39, 2012.
[32] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.
[33] A. E. E.-D. Rashed, H. M. Amer, M. El-Seddek, and H. E.-D. Moustafa, “Sequence Alignment Using Machine Learning-Based Needleman–Wunsch Algorithm,” *IEEE Access*, vol. 9, pp. 109522–109535, 2021, doi: 10.1109/ACCESS.2021.3100408.
[34] M. C. Schatz, C. Trapnell, A. L. Delcher, and A. Varshney, “High-throughput sequence alignment using Graphics Processing Units,” *BMC bioinformatics*, vol. 8, no. 1, pp. 1–10, 2007.

[35] H. L. Chan, T. W. Lam, W. K. Sung, P. W. H. Wong, and S. M. Yiu, “A mutation-sensitive approach for locating conserved gene pairs between related species,” in *Proceedings. Fourth IEEE Symposium on Bioinformatics and Bioengineering*, 2004, pp. 545–552. doi: 10.1109/BIBE.2004.1317390.

[36] P. Borovska, V. Gancheva, and S.-H. Ko, “Scaling of parallel multiple sequence alignment on the supercomputer JUQUEEN,” in *2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*, 2013, pp. 687–691. doi: 10.1109/IDAACS.2013.6663013.

[37] C.-L. Hung, Y.-S. Lin, C.-Y. Lin, Y.-C. Chung, and Y.-F. Chung, “CUDA ClustalW: An efficient parallel algorithm for progressive multiple sequence alignment on Multi-GPUs,” *Computational biology and chemistry*, vol. 58, pp. 62–8, 2015.

[38] K. Katoh and H. Toh, “Recent developments in the MAFFT multiple sequence alignment program. Briefings in Bioinformatics9: 286-298,” 2008.

[39] X. Zhu, K. Li, A. Salah, L. Shi, and K. Li, “Parallel implementation of MAFFT on CUDA-enabled graphics hardware,” *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 12, no. 1, pp. 205–218, 2014.

[40] F. Naznin, R. Sarker, and D. Essam, “Progressive Alignment Method Using Genetic Algorithm for Multiple Sequence Alignment,” *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 615–631, 2012, doi: 10.1109/TEVC.2011.2162849.

[41] C. Shen, B. Liu, and T. Warnow, *eMAFFTadd: scaling MAFFT-linsi-add to large datasets*. 2022. doi: 10.1101/2022.05.23.493139.

[42] K. Tumilaar, Y. Langi, and A. Rindengan, “Hidden Markov Model,” *d’CARTESIAN*, vol. 4, p. 86, Feb. 2015, doi: 10.35799/dc.4.1.2015.8104.

[43] L. Kong, F. Ju, W.-M. Zheng, S. Sun, J. Xu, and D. Bu, *ProALIGN: Directly learning alignments for protein structure prediction via exploiting context-specific alignment motifs*. 2020. doi: 10.1101/2020.12.28.424539.

[44] R. Edgar and K. Sjölander, “SATCHMO: Sequence alignment and tree construction using hidden Markov models,” *Bioinformatics (Oxford, England)*, vol. 19, pp. 1404–11, Aug. 2003, doi: 10.1093/bioinformatics/btg158.

[45] A. Löytynoja and N. Goldman, “WebPRANK: A phylogeny-aware multiple sequence aligner with interactive alignment browser,” *BMC bioinformatics*, vol. 11, p. 579, Nov. 2010, doi: 10.1186/1471-2105-11-579.

[46] E. Meslhy, H. Mousa, and A. Keshk, “E-PROBCONS: Enhanced PROBCONS for Multiple Sequence Alignment,” *IJCI. International Journal of Computers and Information*, Mar. 2020, doi: 10.21608/ijci.2020.15523.1002.

[47] J. Pei and N. Grishin, “MUMMALS: multiple sequence alignment improved by using hidden Markov models with local structural information,” *Nucleic acids research*, vol. 34, pp. 4364–74, Feb. 2006, doi: 10.1093/nar/gkl514.

[48] H. Zhou and Y. Zhou, “SPEM: Improving multiple sequence alignment with sequence profiles and predicted secondary structures,” *Bioinformatics (Oxford, England)*, vol. 21, pp. 3615–21, Sep. 2005, doi: 10.1093/bioinformatics/bti582.

[49] O. O’Sullivan, K. Suhre, C. Abergel, D. Higgins, and C. Notredame, “3DCoffee: Combining Protein Sequences and Structures within Multiple Sequence Alignments,” *Journal of molecular biology*, vol. 340, pp. 385–95, Aug. 2004, doi: 10.1016/j.jmb.2004.04.058.

[50] L. Ait, Z. Yamak, and B. Morgenstern, “DIALIGN at GOBICS—multiple sequence alignment using various sources of external information,” *Nucleic acids research*, vol. 41, Apr. 2013, doi: 10.1093/nar/gkt283.

[51] T. Phuong, C. Do, R. Edgar, and S. Batzoglou, “Multiple alignment of protein sequences with repeats and rearrangements,” *Nucleic acids research*, vol. 34, pp. 5932–42, Feb. 2006, doi: 10.1093/nar/gkl511.
[52] C. Grasso and C. Lee, “Combining Partial Order Alignment and Progressive Multiple Sequence Alignment Increases Alignment Speed and Scalability to Very Large Alignment Problems,” *Bioinformatics (Oxford, England)*, vol. 20, pp. 1546–56, Aug. 2004, doi: 10.1093/bioinformatics/bth126.
[53] W.-K. Sung, *Algorithms in Bioinformatics: A Practical Introduction*, 1st ed. Chapman & Hall/CRC, 2009.
[54] “http://dx.doi.org/10.13140/RG.2.2.27044.96644”.
[55] “Delcher AL, Phillippy A, Carlton J, Salzberg SL. Fast algorithms for large-scale genome alignment and comparison. Nucleic Acids Res. 2002 Jun 1;30(11):2478-83. doi: 10.1093/nar/30.11.2478. PMID: 12034836; PMCID: PMC117189.”.
[56] “Stoye J, Moulton V, Dress AW. DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment. Comput Appl Biosci. 1997;13(6):625-626. doi:10.1093/bioinformatics/13.6.625”.
[57] H. A. Le Thi, T. Pham Dinh, and M. Belghiti, “DCA based algorithms for multiple sequence alignment (MSA),” *Central European Journal of Operations Research*, vol. 22, no. 3, pp. 501–524, Sep. 2014, doi: 10.1007/s10100-013-0324-5.
[58] D. Gusfield, “Efficient methods for multiple sequence alignment with guaranteed error bounds,” *Bulletin of Mathematical Biology*, vol. 55, no. 1, pp. 141–154, Jan. 1993, doi: 10.1007/BF02460299.
[59] P. Borovska and M. Marinova, “Quality of solution of massively parallel MSA based on metaheuristic social behavioral model,” *AIP Conference Proceedings*, vol. 2333. 2021. doi: 10.1063/5.0043739.
[60] “Do CB, Mahabhashyam MS, Brudno M, Batzoglou S. ProbCons: Probabilistic consistency-based multiple sequence alignment. Genome Res. 2005 Feb;15(2):330-40. doi: 10.1101/gr.2821705. PMID: 15687296; PMCID: PMC546535.”.
[61] “Subramanian AR, Kaufmann M, Morgenstern B. DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment. Algorithms Mol Biol. 2008;3:6. Published 2008 May 27. doi:10.1186/1748-7188-3-6”.
[62] S. R. Perumalla and H. Eedi, “Needleman–Wunsch Algorithm Using Multi-threading Approach,” in *Proceedings of the Third International Conference on Computational Intelligence and Informatics*, K. S. Raju, A. Govardhan, B. P. Rani, R. Sridevi, and M. R. Murty, Eds., Singapore: Springer Singapore, 2020, pp. 289–300.
[63] M. A. Alsmirat, Y. Jararweh, M. Al-Ayyoub, M. A. Shehab, and B. B. Gupta, “Accelerating compute intensive medical imaging segmentation algorithms using hybrid CPU-GPU implementations,” *Multimedia Tools and Applications*, vol. 76, pp. 3537–3555, 2017.
[64] F. Gebali, *Algorithms and parallel computing*, vol. 82. John Wiley & Sons, 2011.
[65] M. Farrar, “Striped Smith–Waterman speeds database searches six times over other SIMD implementations,” *Bioinformatics*, vol. 23, no. 2, pp. 156–161, 2007.
[66] T. Rognes, “Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation,” *BMC bioinformatics*, vol. 12, pp. 1–11, 2011.
[67] T. Rognes and E. Seeberg, “Six-fold speed-up of Smith–Waterman sequence database searches using parallel processing on common microprocessors,” *Bioinformatics*, vol. 16, no. 8, pp. 699–706, 2000.
[68] *Tuomanen, B., “Hands-On GPU Programming with Python and CUDA: Explore high-performance parallel computing with CUDA.,” GCTU Repository, accessed September 11, 2024, https://repository.gctu.edu.gh/items/show/1464.*

[69] H. Khaled, R. El Gohary, N. Badr, and H. Faheem, “Hybrid framework for pairwise DNA sequence alignment using the CUDA compatible GPU,” presented at the Proc. Int. Conf. Bioinf. Comput. Biol.(BIOCOMP), 2013, p. 1.
[70] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Pearson Education, 2010. [Online]. Available: https://books.google.bg/books?id=49OmnOmTEtQC
[71] K. Balhaf, M. A. Alsmirat, M. Al-Ayyoub, Y. Jararweh, and M. A. Shehab, “Accelerating Levenshtein and Damerau edit distance algorithms using GPU with unified memory,” presented at the 2017 8th international conference on information and communication systems (ICICS), IEEE, 2017, pp. 7–11.
[72] G. Barlas, *Multicore and GPU programming : an integrated approach*, 1 online resource vols. Amsterdam: Morgan Kaufmann, 2015. [Online]. Available: https://www.overdrive.com/search?q=7AF769E6-89B4-42C0-B364-4D51449C41C9
[73] J. Pevsner, *Bioinformatics and Functional Genomics*. Wiley, 2015. [Online]. Available: https://books.google.bg/books?id=OaRjCgAAQBAJ
[74] M. Agostino, *Practical Bioinformatics*. CRC Press, 2012. [Online]. Available: https://books.google.bg/books?id=RQcPBAAAQBAJ
[75] C. Orengo, D. T. Jones, and J. M. Thornton, *Bioinformatics: Genes, Proteins and Computers*. in Advanced text. BIOS Scientific, 2003. [Online]. Available: https://books.google.bg/books?id=9kslngEACAAJ
[76] S. Mallick and D. Reich, “The Allen Ancient DNA Resource (AADR): A curated compendium of ancient human genomes.” Harvard Dataverse, 2023. doi: 10.7910/DVN/FFIDCW.
[77] “Landrum MJ, Lee JM, Benson M, Brown GR, Chao C, Chitipiralla S, Gu B, Hart J, Hoffman D, Jang W, Karapetyan K, Katz K, Liu C, Maddipatla Z, Malheiro A, McDaniel K, Ovetsky M, Riley G, Zhou G, Holmes JB, Kattman BL, Maglott DR. ClinVar: improving access to variant interpretations and supporting evidence. Nucleic Acids Res. 2018 Jan 4;46(D1):D1062-D1067. doi: 10.1093/nar/gkx1153. PMID: 29165669; PMCID: PMC5753237.”.
[78] J. Piñero *et al.*, “The DisGeNET knowledge platform for disease genomics: 2019 update,” *Nucleic Acids Research*, vol. 48, no. D1, pp. D845–D855, Nov. 2019, doi: 10.1093/nar/gkz1021.
[79] D. Toncheva, S. Karachanak-Yankova, M. Marinova, P. Borovska, and D. Serbezov, “Susceptibility to Neurodegenerative Disorders: Insights from Paleogenomic Data,” *Human Biology*, vol. 93, no. 4, pp. 289–297, 2024, doi: 10.1353/hub.2021.a917652.
[80] S. N. Devi and S. Rajagopalan, “An index based pattern matching using multithreading,” *International Journal of Computer Applications*, vol. 50, no. 6, 2012.
[81] M. I. Irawan, I. Mukhlash, A. Rizky, and A. R. Dewi, “Application of Needleman-Wunch Algorithm to identify mutation in DNA sequences of Corona virus,” *Journal of Physics: Conference Series*, vol. 1218, no. 1, p. 012031, May 2019, doi: 10.1088/1742-6596/1218/1/012031.
[82] A. Chaudhary, D. Kagathara, and V. Patel, “A gpu based implementation of needleman-wunsch algorithm using skewing transformation,” presented at the 2015 Eighth International Conference on Contemporary Computing (IC3), IEEE, 2015, pp. 498–502.
[83] L. Ji, X. Pu, H. Qu, and G. Liu, “One-dimensional pairwise CNN for the global alignment of two DNA sequences,” *Neurocomputing*, vol. 149, pp. 505–514, 2015.
[84] J. Brownlee, *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python*. Machine Learning Mastery, 2020.
[85] D. Conway and J. M. White, *Machine learning for hackers*. Sebastopol, CA: O’Reilly, 2012.
[86] M. Sewak, Md. R. Karim, and P. Pujari, *Practical Convolutional Neural Networks: Implement advanced deep learning models using Python*. Packt Publishing, 2018.
[87] A. Vaswani *et al.*, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett,

Eds., Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[88] T. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," *ArXiv*, vol. abs/1508.04025, 2015, [Online]. Available: https://api.semanticscholar.org/CorpusID:1998416

[89] Y. S. Lee, Y. S. Kim, and R. L. Uy, "Serial and parallel implementation of Needleman-Wunsch algorithm.," *International Journal of Advances in Intelligent Informatics*, vol. 6, no. 1, 2020.

[90] https://docs.nvidia.com/cuda/

[91] https://www.techpowerup.com/review/evga-geforce-rtx-3060-xc/2.html