DOI 10.37882/2223-2966.2025.05-2.13

РАСПРЕДЕЛЁННАЯ СИСТЕМА ДЛЯ ПОСТРОЕНИЯ МНОЖЕСТВЕННЫХ ВЫРАВНИВАНИЙ СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ МЕТОДОМ MAHDS

DISTRIBUTED SYSTEM FOR MULTIPLE SEQUENCE ALIGNMENT BY MEANS OF MAHDS METHOD

D. Kostenko
E. Korotkov
M. Korotkova

Summary. The search for the optimal multiple sequence alignment requires a large amount of computing resources. Previously, we developed the MAHDS method, which is capable of constructing statistically significant alignments of weakly homologous sequences. This paper presents the architecture and some details of the software implementation of the system for constructing multiple alignments using this method. An effective hierarchical approach used for distributing computations in an MPI cluster environment is described. An approach is proposed to ensure secure access of the end users of the system to the computing cluster resources.

Keywords: multiple sequence alignment, distributed computing, computing cluster, MPI, thread pool.

Костенко Дмитрий Олегович
Аспирант, младший научный сотрудник, Национальный исследовательский ядерный университет «МИФИ» (Москва); Федеральный исследовательский центр «Фундаментальные основы биотехнологии» РАН (Москва)
dk0stenko@yandex.ru
Коротков Евгений Вадимович
Доктор биологических наук, профессор, Национальный исследовательский ядерный университет «МИФИ» (Москва); Федеральный исследовательский центр «Фундаментальные основы биотехнологии» РАН (Москва)
bioinf@yandex.ru
Короткова Мария Александровна
Кандидат технических наук, доцент, Национальный исследовательский ядерный университет «МИФИ» (Москва)
discretmath@gmail.com

Аннотация. Поиск оптимального множественного выравнивания символьных последовательностей требует затраты большого объёма вычислительных ресурсов. Ранее нами был разработан метод MAHDS, способный строить статистически значимые выравнивания слабо гомологичных последовательностей. В данной работе представлена архитектура и некоторые детали программной реализации системы для построения множественных выравниваний этим методом. Описан эффективный иерархический подход, использующийся для распределения вычислений в среде MPI кластера. Предложен способ обеспечения безопасного доступа конечного пользователя системы к ресурсам вычислительного кластера.

Ключевые слова: множественное выравнивание, распределённые вычисления, вычислительный кластер, MPI, thread pool.

Введение

Множественные выравнивания символьных последовательностей (MSA) могут применяться для сравнения разнообразных объектов, представимых в виде символьных последовательностей с произвольным алфавитом A. Исходное множество последовательностей обозначим S, его элементы: $S_1, S_2, \ldots, S_n$, а их длины соответственно $L_1, L_2, \ldots, L_n$. Предположим, что последовательности $S_1, S_2, \ldots, S_n$ (или отдельные их участки) получены из одной неизвестной первоначальной последовательности S_{par} длины L путём внесения в неё мутаций: замен, вставок и удалений (делеций) символов. Задача построения MSA состоит в нахождении мутаций в исследуемых последовательностях, которые привели к их расхождению. Последовательности $S_1, S_2, \ldots, S_n$ при этом дополняются вхождениями специального символа «–», который ставится в соответствие удалённым в данной последовательности символам или вставленным символам в другой последовательности. Итоговое выравнивание представлено дополненными исходными последовательностями, длины которых ста-

новятся одинаковыми (обозначим длину выравнивания $L_a \geq L$).

Можно найти разные практические приложения задачи MSA, например, в [1] описано применение к валютным рядам подхода, использующегося для построения выравниваний. Однако основной областью применения MSA является биоинформатика. Первичные структуры ДНК/РНК и белков представляются символьными последовательностями с алфавитами, включающими в себя 4 и 20 символов соответственно. MSA позволяет обнаруживать структурные и функциональные взаимосвязи таких последовательностей и используется для решения многих задач биоинформатики. В частности, MSA применяется в работе модели AlphaFold [2,3], предсказывающей структуры белков, за разработку которой была присуждена Нобелевская премия по химии в 2024 году.

Несмотря на практическую значимость MSA, на данный момент не был разработан идеальный метод решения этой задачи. Это связано с тем, что задача построения множественного выравнивания является NP-полной в большинстве постановок [4], следовательно алгоритм её решения имеет экспоненциальную сложность, неприемлемую для применения на практике. В биоинформатике для построения множественных выравниваний применяются различные эвристические алгоритмы, имеющие сложность не более полиномиальной [5–9]. Однако даже при сравнительно хорошей асимптотике использующихся алгоритмов, решение реальных задач обычно сопряжено с распределёнными вычислениями, так как объём обрабатываемых данных может быть очень велик.

Ранее нами был разработан метод MAHDS, отличительной особенностью которого является способность строить статистически значимые выравнивания последовательностей, накопивших большое количество мутаций [10]. Если зафиксировать максимальную длину входных последовательностей и в качестве переменной рассматривать количество последовательностей n, тогда алгоритм MAHDS будет иметь временную и пространственную сложность $O(n)$. Если же наоборот зафиксировать количество последовательностей и варьировать их максимальную длину m, тогда алгоритм обладает сложностью $O(m^2)$, таким образом можно считать, что алгоритм имеет асимптотику $O(d^2)$, где d — объём входных данных [11]. По определению это означает, что $\exists C \in (0,+\infty) \exists d_0 \forall d : d > d_0 f(d) \leq Cd^2$, где $f(d)$ — функция зависимости времени работы алгоритма (или количества элементарных операций) от объёма входных данных, C — положительная константа. Помимо большого количества и длины последовательностей, характерных для реальных биоинформатических приложений, порядок величины C обуславливает необходимость в существенном объёме вычислительных ресурсов для решения задачи MSA за приемлемое время. Поэтому при разработке программной реализации MAHDS особое внимание уделялось эффективности использования вычислительных ресурсов и обеспечению горизонтальной масштабируемости на узлах вычислительного MPI кластера.

Алгоритм MAHDS

В рамках данной статьи будет дано только краткое описание MAHDS с акцентом на аспектах, важных для программной реализации. Подробное описание метода представлено в работах [10,11].

Идея MAHDS состоит в сведении задачи построения оптимального выравнивания последовательностей из множества S к задаче оптимизации позиционно-весовых матриц (PWM). PWM — это матрица размерности $|A| \times L$, строки которой соответствуют символам исходного алфавита A, а столбцы — столбцам в MSA без учёта вставок (такие позиции в выравнивании также соответствуют символам предполагаемого «общего предка» S_{par}). Для удобства будем кодировать символы алфавита A целыми числами на отрезке $[1,|A|]$. Чем больше величина элемента $PWM(i,j)$, тем больше вероятность, что $S_{par}(j) = i$, и тем больше поощряется постановка i в j-тую позицию MSA.

Используя в качестве входных данных произвольную PWM и множество последовательностей S можно построить MSA. Для этого с использованием подхода, подобного алгоритму Нидлмана-Вунша [12] строятся парные выравнивания каждой последовательности S_i ($i = 1..n$). Все последовательности из S выравниваются со специальной последовательностью S_a вида $1,2,\ldots,L$, символы которой соответствуют позициям в MSA, а в качестве матрицы замен [13] используется PWM. Полученные парные выравнивания накладываются таким образом чтобы одинаковые символы выровненных S_a попали в одни и те же столбцы, после чего S_a удаляется и остаётся MSA. При построении каждого парного выравнивания вычисляется значение функции сходства F_i ($i = 1..n$). Показателем качества PWM и соответствующего MSA является сумма этих величин $FS = \sum_{i=1}^{n} F_i$.

По известному MSA достаточно просто восстановить PWM, в некоторой мере подходящую для выравнивания данных последовательностей. Для этого по MSA рассчитывается частотная матрица, которая далее нормализуется как показано в [14], после чего может рассматриваться в качестве PWM. Такое преобразование позволяет итеративно оптимизировать PWM для построения наи-

лучшего MSA. На каждой итерации PWM применяется для построения MSA, после чего по полученному выравниванию восстанавливается новая версия PWM, лучше подстроенная под последовательности *S*. Целевой функцией оптимизации является *FS*, а критерием останова — прекращение роста целевой функции.

Эмпирически было проверено, что итеративная процедура практически всегда сходится менее чем за 10 итераций, поэтому их количество можно считать константой. Однако при оптимизации PWM имеется тенденция к попаданию в локальные максимумы, поэтому для нахождения наилучшей PWM сначала генерируется множество *Q* случайных инициализирующих матриц той же размерности, что и PWM (по умолчанию $|Q| = 400$). Эти матрицы рассматриваются как точки в пространстве размерности $L * |A|$, и процедура их генерации максимизирует минимальное Евклидово расстояние между матрицами. К каждой инициализирующей матрице из множества *Q* после нормализации применяется итеративная процедура оптимизации; PWM с наибольшим значением *FS* рассматривается как оптимальная и применяется для построения результирующего MSA.

Статистическая значимость MSA_0 и соответствующей PWM_0 оценивается по методу Монте-Карло, предложенному в [15]. Для этого для этого символы последовательностей из множества *S* многократно перемешиваются и полученные последовательности выравниваются с помощью PWM_0. Статистическая значимость вычисляется по формуле $Z = \frac{FS_0 - \overline{FS}}{\sigma FS}$, где $\overline{FS}$ — оценка математического ожидания *FS*, σFS — оценка среднеквадратичного отклонения *FS* для перемешанных *S*.

Оптимальная величина *L* изначально неизвестна. Поэтому для нахождения наиболее оптимального MSA мы вынуждены перебирать множество L_s значений *L*. Здесь критерием наилучшего MSA является наибольшая статистическая значимость *Z*, так как величина *FS* при разных *L* может масштабироваться. Обычно оптимальное значение *L* близко к средней длине выравниваемых последовательностей $\bar{L} = \left\lfloor \frac{\sum_{i=1}^{n} L_i}{n} \right\rfloor$. Поэтому наиболее осмысленный вид множества:

$$L_s = \{L | (\bar{L} - \Delta L \le L \le \bar{L} + \Delta L) \& (L \bmod s = 0)\},$$

где ΔL — максимальное отклонение от среднего значения, *s* — шаг.

Параллельные вычисления в программной реализация MAHDS

Из представленного описания алгоритма видно, что его параллельное исполнение может быть организовано на нескольких уровнях вложенности при выполнении следующих процедур: вычисление наилучших PWM при разных значениях *L*; выравнивание множеств перемешанных последовательностей при вычислении *Z*; итеративная оптимизация матриц из множества *Q*; Построение парных выравниваний для каждой последовательности из множества *S*; двумерное динамическое программирование при построении парных выравниваний. Для минимизации затрат вычислительных ресурсов, затрачиваемых непосредственно на организацию параллельных вычислений, использовались только два верхних уровня вложенности.

Программная реализация MAHDS изначально разрабатывалась в виде консольного MPI приложения для CPU кластеров. Использование MPI продиктовано исключительно его распространённостью и, во многих случаях, безальтернативностью в контексте применения вычислительных кластеров научных и образовательных организаций. Программный код написан на языке C++ для достижения наибольшей производительности.

Интенсивный обмен данными между MPI процессами, работающими на разных узлах кластера, требует передачи этих данных по сети, что может существенно замедлять вычисления. По этой причине мы минимизируем количество MPI процессов и распределяем между ними только максимально независимые вычисления самого верхнего уровня вложенности. Мы создаём только по одному MPI процессу на каждый доступный узел кластера. В свою очередь, каждый такой процесс применяет потоки операционной системы для эффективного использования ресурсов узла. Таким образом, кроме иерархичности (вложенности) процедур, подходящих для параллельного исполнения, естественным образом возникает и иерархичность исполнителей. Ниже представлена схема их соответствия (см. рис. 1).

На представленной схеме ромбами обозначены распараллеливающиеся циклы. Кругами обозначены точки синхронизации. Длинным пунктиром обозначено использование MPI процессов для распределения вычислений. Линией из точек обозначено параллельное исполнение с помощью потоков операционной системы. Коротким пунктиром обозначено комбинированное применение обоих способов распараллеливания для предварительной генерации инициализирующих матриц. Свойства множества инициализирующих матриц зависят только от мощности алфавита *A* и величины *L*, поэтому матрицы бессрочно кешируются в базе данных для повторного использования при следующих запусках программы.

Нельзя точно предсказать, сколько времени займёт итеративная процедура оптимизации PWM полученной из инициализирующей матрицы. По этой причине для

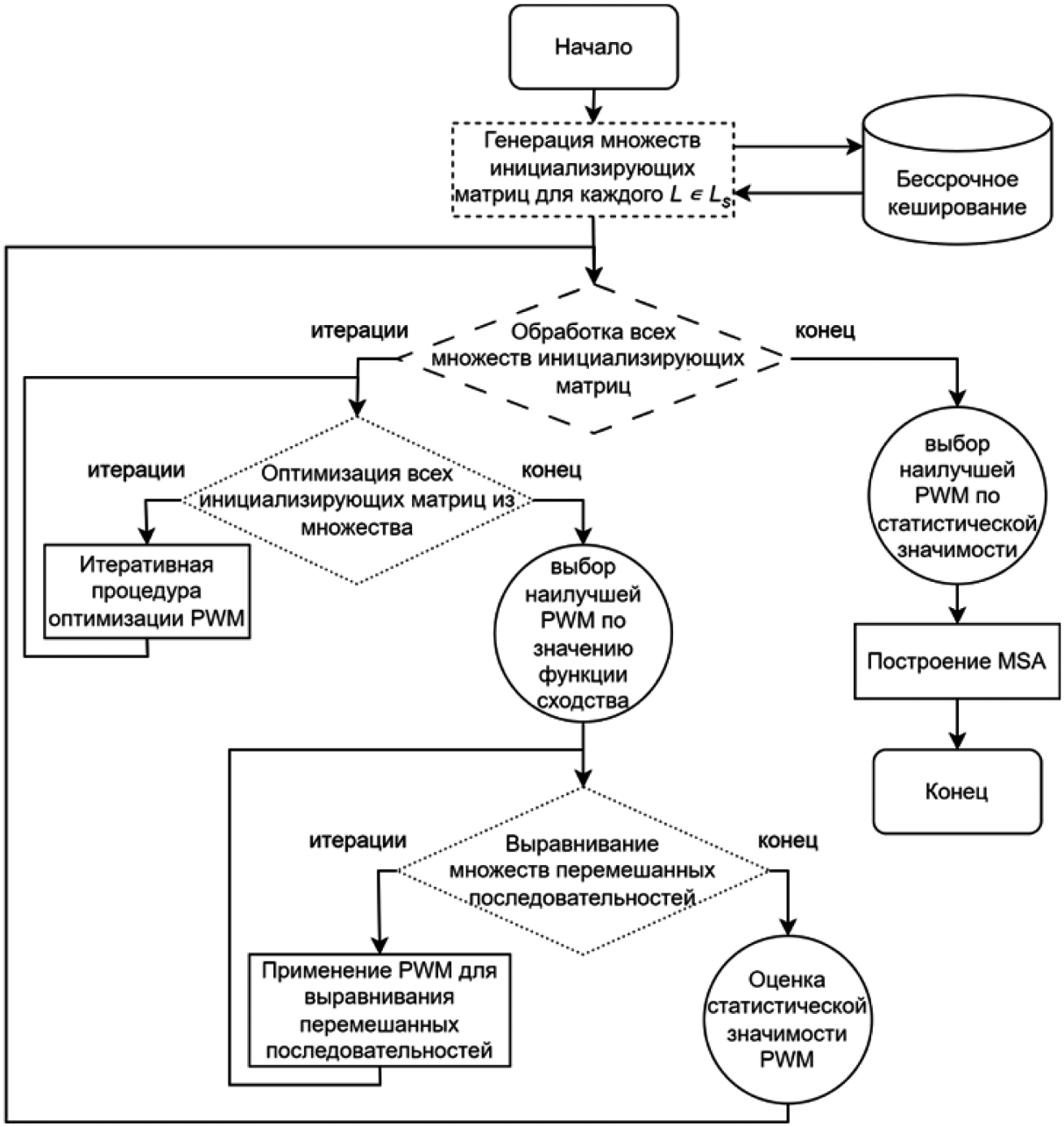

Рис. 1. Параллельное исполнение MAHDS

того, чтобы большую часть времени все физические ядра вычислительных узлов были заняты полезными вычислениями, мы применяем пул потоков (thread pool) [16]. Задачи на оптимизацию каждой инициализирующей матрицы помещаются в очередь, а «работники», закреплённые за потоками, по готовности их извлекают из очереди и исполняют.

Архитектура программной системы для построения выравниваний

Для того чтобы позволить конечному пользователю, не имеющему доступа к какому-либо вычислительному кластеру, строить множественные выравнивания достаточно быстро, была разработана программная система, развёрнутая в данный момент на кластере группы математического анализа последовательностей ДНК и белков ФИЦ Биотехнологии РАН. Система предоставляет возможность строить и оценивать множественные выравнивания через веб-интерфейс, не имея непосредственного доступа к кластеру. Пользователь может послать одновременно ограниченное количество задач на исполнение, после чего каждая задача будет некоторое время (от десятков секунд до десятков часов в зависимости от количества и длины последовательностей) обрабатываться, и результат вычислений будет доступен в течение недели после его получения. Запуск задач доступен только для зарегистрированных авторизованных

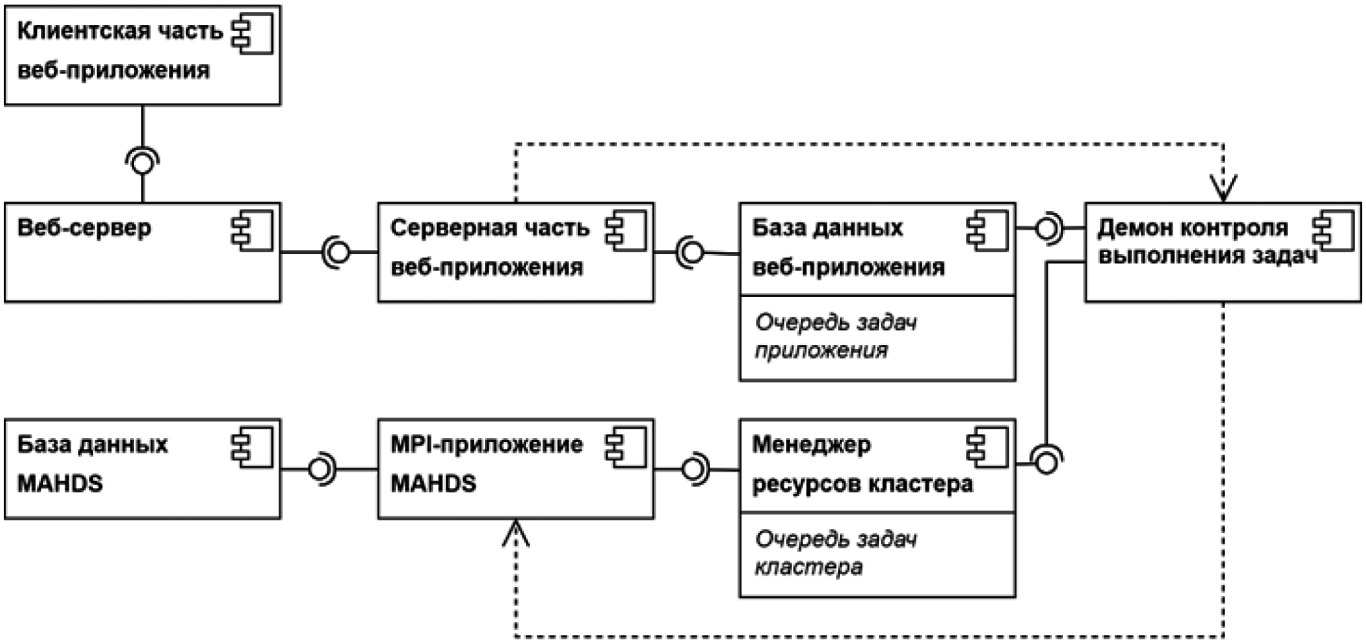

Рис. 2. UML диаграмма компонентов системы для построения множественных выравниваний методом MAHDS

пользователей с подтверждённым адресом электронной почты. Выше представлена UML диаграмма компонентов системы (см. рис. 2).

При добавлении задачи пользователем, запрос с клиентской части веб-приложения (написана на Vue 3) отправляется по протоколу HTTP на веб-сервер (в нашем случае Apache). Серверная часть веб-приложения реализована на Python с использованием фреймворка Flask, поэтому стандартом взаимодействия между Apache и приложением является WSGI. Серверная часть веб-приложения предоставляет REST API и не хранит состояние, из-за чего непосредственно в рамках данного компонента крайне сложно контролировать выполнение задач на кластере. В нашей системе эту функцию берёт на себя демон (служба) контроля выполнения задач (написан на Python и добавлен в systemd). Если запрос, отправленный клиентом, проходит валидацию на сервере, задача записывается в базу данных (используется СУБД SQLite) и проверяется состояние демона (при необходимости он запускается), а клиенту возвращается ссылка на страницу, где будет размещён результат.

Демон контроля выполнения задач постоянно работает в фоновом режиме и извлекает задачи из базы данных в порядке их добавления, готовит всё необходимое окружение для работы программы, реализующей MAHDS, и ставит задачи в очередь кластера с помощью менеджера ресурсов кластера (в нашем случае Slurm). Также демон контроля выполнения задач отвечает за ожидание результатов вычислений и погружение этих результатов в базу данных веб-приложения. Кроме того, в задачи демона входит удаление старых результатов и ограничение объёма ресурсов, потребляемых системой для построения выравниваний (та же самая очередь задач кластера может использоваться параллельно и для других задач). Особое внимание при реализации данного компонента уделялось обработке любых возможных ошибок, так как демон контроля выполнения задач должен работать в бесконечном цикле без сбоев.

Расположение компонентов на диаграмме, представленной на рисунке 2, отражает особенности развёртывания программной системы. Первая строка соответствует устройству клиента, через которое осуществляется доступ к системе. На второй строке представлены компоненты, развёрнутые на веб-сервере. Компоненты на третьей строке развёрнуты на управляющем и вычислительных узлах кластера в общем дисковом пространстве, реализованном с помощью NFS.

Обсуждение и результаты

Нами была разработана распределённая программная система для построения множественных выравниваний методом MAHDS. Был предложен иерархический подход к организации параллельных вычислений, который позволил минимизировать объём передаваемых по сети данных, сократить время блокирования на примитивах синхронизации и обеспечить эффективное использование всех доступных вычислительных ресурсов. Для обеспечения безопасного доступа конечных пользователей к программной реализации MAHDS, развёрнутой на вычислительном кластере, было разработано веб-приложение (доступно по ссылке http://victoria.biengi.ac.ru/mahds) и демон, контролирующий выполнение задач. Задачи пользователей попадают сначала в очередь задач приложения, после чего, согласно своей

конфигурации, демон добавляет их в очередь задач кластера с ограничениями по доступным ресурсам. Таким образом работа системы для построения выравниваний не мешает использованию той же самой очереди задач кластера для других целей.

При разработке программной реализации MAHDS приоритетной целью было обеспечение максимальной производительности. Для других компонентов системы такая задача не стояла, так как построение выравниваний является бутылочным горлышком системы.

Вычисления в разных областях науки часто сопряжены с использованием CPU кластеров. Представленные в данной работе решения могут применяться для разработки доступных вычислительных инструментов для учёных и инженеров.

ЛИТЕРАТУРА

1. Коротков Е.В., Короткова М.А. Разработка математического метода для поиска скрытой периодичности в аминокислотных последовательностях белков с учётом делеций и вставок: 6 // Биофизика. 2015. Vol. 60, № 6. P. 1057–1068.
2. Jumper J. et al. Highly accurate protein structure prediction with AlphaFold: 7873 // Nature. Nature Publishing Group, 2021. Vol. 596, № 7873. P. 583–589.
3. Abramson J. et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3 // Nature. Nature Publishing Group, 2024. Vol. 630, № 8016. P. 493–500.
4. Wang L., Jiang T. On the complexity of multiple sequence alignment.: 4 // J Comput Biol. United States, 1994. Vol. 1, № 4. P. 337–348.
5. Katoh K. et al. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform // Nucleic Acids Res. 2002. Vol. 30, № 14. P. 3059–3066.
6. Edgar R.C. MUSCLE: multiple sequence alignment with high accuracy and high throughput // Nucleic Acids Res. 2004. Vol. 32, № 5. P. 1792–1797.
7. Notredame C., Higgins D.G., Heringa J. T-coffee: A novel method for fast and accurate multiple sequence alignment: 1 // Journal of Molecular Biology. 2000. Vol. 302, № 1. P. 205–217.
8. Sievers F., Higgins D.G. Clustal Omega for making accurate alignments of many protein sequences // Protein Science. 2018. Vol. 27, № 1. P. 135–145.
9. Lassmann T., Frings O., Sonnhammer E.L.L. Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features // Nucleic Acids Res. 2009. Vol. 37, № 3. P. 858–865.
10. Korotkov E.V. et al. Multiple Alignment of Promoter Sequences from the Arabidopsis thaliana L. Genome: 2 // Genes. 2021. Vol. 12, № 2. P. 135.
11. Korotkov E.V., Kostenko D.O. Application of the MAHDS Method for Multiple Alignment of Highly Diverged Amino Acid Sequences: 7 // International Journal of Molecular Sciences. 2022. Vol. 23, № 7.
12. Needleman S.B., Wunsch C.D. A general method applicable to the search for similarities in the amino acid sequence of two proteins: 3 // Journal of Molecular Biology. Academic Press, 1970. Vol. 48, № 3. P. 443–453.
13. Mount D.W. Comparison of the PAM and BLOSUM Amino Acid Substitution Matrices // Cold Spring Harb Protoc. Cold Spring Harbor Laboratory Press, 2008. Vol. 2008, № 6. P. pdb.ip59.
14. Pugacheva V., Korotkov A., Korotkov E. Search of latent periodicity in amino acid sequences by means of genetic algorithm and dynamic programming: 5 // Statistical Applications in Genetics and Molecular Biology. 2016. Vol. 15, № 5.
15. Comet J.P. et al. Significance of Z-value statistics of Smith-Waterman scores for protein alignments: 3–4 // Comput Chem. 1999. Vol. 23, № 3–4. P. 317–331.
16. Shoshany B. A C++17 thread pool for high-performance scientific computing // SoftwareX. 2024. Vol. 26. P. 101687.

Журнал «Современная наука: актуальные проблемы теории и практики»