ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2025 Управление, вычислительная техника и информатика № 72
Tomsk State University Journal of Control and Computer Science

Научная статья
УДК 004.021
doi: 10.17223/19988605/72/7

Алгоритм высокопроизводительного обнаружения регионов низкой сложности в длинных геномных последовательностях

Ростислав Сергеевич Воробьёв[1], Александр Владимирович Замятин[2], Татьяна Сергеевна Геращенко[3], Анастасия Алексеевна Коробейникова[5], Евгений Владимирович Денисов[5]

[1, 3, 4, 5] Научно-исследовательский институт онкологии Томского национального исследовательского медицинского центра Российской академии наук, Томск, Россия
[1, 2] Национальный исследовательский Томский государственный университет, Томск, Россия
[1] tsu@rvorobev.ru
[2] zamyatin@mail.tsu.ru
[3] t_gerashchenko@oncology.tomsk.ru
[4] shegolmay@gmail.com
[5] d_evgeniy@oncology.tomsk.ru

Аннотация. Обнаружение регионов низкой сложности (Low Complexity Regions, LCR) в геномных последовательностях представляет важную задачу для множества биоинформационных инструментов, включая выравнивание последовательностей, дизайн зондов и обнаружение вариантов. В настоящей работе представлен DUSTSCAN – инструмент и модификация алгоритма DUST (оценка распределения частот уникальных триплетов в последовательности) для идентификации LCR с применением параллельных вычислений для значительного ускорения расчетов. Проводится сравнительный анализ DUSTSCAN с другими версиями алгоритма DUST. Результаты показывают значительный прирост в скорости обнаружения регионов низкой сложности.

Ключевые слова: алгоритм; параллельные вычисления; регионы низкой сложности.

***Благодарности*:** Работа выполнена в рамках государственного задания Министерства науки и высшего образования РФ № 075-00490-25-04 (регистрационный номер 125042105351-3).

***Для цитирования*:** Воробьёв Р.С., Замятин А.В., Геращенко Т.С., Коробейникова А.А., Денисов Е.В. Алгоритм высокопроизводительного обнаружения регионов низкой сложности в длинных геномных последовательностях // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2025. № 72. С. 71–79. doi: 10.17223/19988605/72/7

Original article
doi: 10.17223/19988605/72/7

A high-performance algorithm for detecting low-complexity regions in long genomic sequences

Rostislav S. Vorobev[1], Alexander V. Zamyatin[2], Tatiana S. Gerashchenko[3], Anastasia A. Korobeynikova[4], Evgeny V. Denisov[5]

[1, 3, 4, 5] Cancer Research Institute, Tomsk National Research Medical Center, Russian Academy of Sciences, Tomsk, Russian Federation
[1, 2] National Research Tomsk State University, Tomsk, Russian Federation
[1] tsu@rvorobev.ru
[2] zamyatin@mail.tsu.ru
[3] t_gerashchenko@oncology.tomsk.ru
[4] shegolmay@gmail.com
[5] d_evgeniy@oncology.tomsk.ru

Abstract. Detection of Low Complexity Regions (LCR) in genomic sequences is a crucial task for numerous bioinformatics tools, including sequences alignment, probes design, variants calling. This study introduces DUSTSCAN as a modification of the DUST algorithm (score estimation of frequencies distributiion of unique triplets in a sequence) for identifying Low Complexity Regions, utilizing parallel computing to significantly accelerate calculations. This research presents a comparative analysis of DUSTSCAN with other versions of the DUST algorithm. The results demonstrate a significant improvement in detection speed, making the new approach particularly valuable for large-scale genomic data processing tasks. The developed tool can be effectively applied in various bioinformatics pipelines, enhancing the performance of tasks that require LCR identification in genomic sequences.

Keywords: algorithm; parallel computing; low complexity regions.

***Acknowledgments*:** The study was carried out according to the state assignment of the Ministry of Science and Higher Education of the Russian Federation № 075-00490-25-04 (Registration number 125042105351-3).

***For citation*:** Vorobev, R.S., Zamyatin, A.V., Gerashchenko, T.S., Korobeynikova, A.A., Denisov, E.V. (2025) A high-performance algorithm for detecting low-complexity regions in long genomic sequences. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitelnaja tehnika i informatika – Tomsk State University Journal of Control and Computer Science*. 72. pp. 71–79. doi: 10.17223/19988605/72/7

Введение

Геномные последовательности содержат регионы с различной степенью сложности и вариабельности [1]. Регионы низкой сложности (Low Complexity Regions, LCR) – участки геномных последовательностей с повторяющимися или смещенными нуклеотидными композициями – представляют особый интерес и вызов для множества биоинформационных методов [2–4]. Эти регионы могут значительно затруднять работу алгоритмов выравнивания последовательностей [5], влиять на обнаружение структурных и точечных вариантов [6–8], ухудшать качество сборки геномов и транскриптомов, а также снижать специфичность зондов для ДНК-гибридизации [9–11].

Для идентификации регионов низкой сложности в геномных последовательностях существует несколько различных подходов. Классический алгоритм DUST, основанный на анализе частот *k*-меров, изначально разработан для анализа белковых последовательностей, но затем был адаптирован и для анализа нуклеотидных последовательностей [12]. Позднее A. Morgulis и соавт. представили симметричную реализацию алгоритма, известную как SDUST, которая эффективнее с вычислительной точки зрения [2].

Для выявления участков низкой сложности в геномных последовательностях могут использоваться два взаимодополняющих алгоритма: классический DUST и его симметричная модификация SDUST [2].

Алгоритм DUST основан на анализе распределения триплетов (*k*-меров размера 3) нуклеотидов в скользящих окнах фиксированной длины. Для окна *w* значение сложности определяется как

$$DUST(w) = \frac{1}{w-2}\sum_{i=1}^{64}\frac{c_i(c_i-1)}{2}, \quad (1)$$

где c_i – частота *i*-го триплета в рассматриваемом окне (например, разные триплеты AAA и AAT).

Несмотря на широкое применение, оригинальный алгоритм DUST (1) обладает существенными недостатками ассимметричности относительно обратной последовательности генома и контекстной зависимостью, что означает, что результат маскирования подпоследовательности может зависеть от ее окружения.

Для преодоления ограничений классического подхода был разработан алгоритм SDUST (Symmetric DUST), обладающий улучшенными характеристиками при сохранении базового принципа оценки комплексности.

Для подпоследовательности длиной *w* оценка сложности определяется как

$$SDUST(w) = \frac{1}{w-2}\sum_{i=1}^{32}\frac{c_i^{sym}(c_i^{sym}-1)}{2}, \quad (2)$$

где c_i^{sym} – частота вхождений симметричного триплета (учитывается как прямая, так и обратно-комплементарная форма записи; например, AAT и его обратно-комплементарная форма ATT).

Алгоритм SDUST (2) превосходит оригинальный DUST по качеству маскирования [2].

Значения алгоритма DUST лишь демонстрируют оценку сложности некой последовательности и затем могут быть использованы для выделения низко-сложных участков в геномной последовательности. Для выделения непосредственно самих регионов низкой сложности с начальными и конечными координатами используется пороговое значение. Удовлетворяющие пороговому значению оценки образуют LCR, координаты которого рассчитываются, исходя из относительных позиций в геномной последовательности.

Помимо подходов, основанных на частотах *k*-меров, существуют методы идентификации LCR на основе энтропии Шеннона [3, 13], лингвистической сложности [4], и другие специализированные алгоритмы. Например, программа RepeatMasker использует подход, основанный на выравнивании для маскирования повторяющихся элементов [14], а fastqc и fastp включают инструменты для оценки чрезмерной представленности *k*-меров в данных секвенирования [15, 16].

С ростом объемов геномных данных, особенно с появлением технологий высокопроизводительного секвенирования, возникает необходимость в эффективных алгоритмах для анализа LCR в больших датасетах. Графические процессоры (GPU) предоставляют возможность значительного ускорения вычислений благодаря массивному параллелизму [17–21]. Вычисления общего назначения на графических процессорах (GPGPU) успешно применяются в различных биоинформационных задачах, таких как выравнивание последовательностей, поиск сходства, предсказание структуры белков, анализ транскриптомов [20].

Однако многие существующие инструменты на основании имеющихся модификаций алгоритма DUST все еще требуют значительного времени для обработки длинных геномных последовательностей, что особенно заметно при обработке больших геномных последовательностей.

1. Постановка задачи

В данной работе предлагается модификация DUSTSCAN, которая включает в себя ускорение обнаружения регионов низкой сложности за счет распараллеливания вычислений.

Основная идея алгоритма заключается в параллельной обработке множества регионов исходной геномной последовательности с применением скользящих окон для анализа частот триплетов и дополнительной пост-обработки для объединения результатов и выявления начальных и конечных позиций регионов. В отличие от классических реализаций алгоритма DUST, которые оптимизированы для последовательной обработки, DUSTSCAN использует массивный параллелизм для одновременной обработки множества окон и определения координат границ регионов. Каждая позиция в геномной последовательности является потенциальной начальной точкой для скользящего окна. Каждая исходная последовательность разбивается на регионы, для каждого из которых параллельно вычисляется $DUST(w)$. За счет задания фиксированного порогового значения выделяются позиции-кандидаты для формирования регионов низкой сложности, которые на этапе пост-обработки фильтруются с помощью порогового значения и объединяются во множество начальных и конечных координат соответствующих регионов в рамках координат исходных геномных последовательностей (рис. 1).

Для исходной последовательности G производится разбиение на k перекрывающихся фрагментов C_i размером M:

$$C_i = G\left[i \cdot M : (i+1) \cdot M + W - 1\right], \text{ при } \; i = 0,1,...,k-1, \tag{3}$$

где W – размер окна анализа.

Для каждого чанка C_i параллельно вычисляется множество локальных регионов R_i^{local} с использованием DUST:

$$R_i^{local} = \left\{ r_j^{local} = \left(s_j^{local}, e_j^{local}, score_j \right) : DUST(w_i) > \theta \right\}, \tag{4}$$

где s_j^{local} и e_j^{local} – локальные координаты начала и конца j-го региона в чанке C_i, а θ – пороговое значение оценки.

Глобальные координаты регионов определяются через преобразование

$$R_i^{global} = \left\{ \left(s_j^{local} + i \cdot M, e_j^{local} + i \cdot M, score_j \right) : \left(s_j^{local}, e_j^{local}, score_j \right) \in R_i^{local} \right\} \quad (5)$$

Рис. 1. Концептуальная схема параллельной обработки в DUSTSCAN (R1, R2 – обнаруженные регионы)
Fig. 1. Conceptual scheme of parallel processing in DUSTSCAN (R1, R2 – detected regions)

Финальное множество регионов формируется объединением и последующим слиянием перекрывающихся участков:

$$R_{final} = merge(\bigcup_{i=0}^{k-1} R_i^{global}, d_{merge}), \quad (6)$$

где d_{merge} – максимальное расстояние для слияния соседних регионов.

Данный подход обеспечивает масштабируемость вычислений и позволяет эффективно использовать как многоядерные CPU, так и массивно-параллельные GPU-архитектуры.

2. Параметры тестовой среды

Проведена оценка скорости выполнения инструмента *dustscan* с существующими инструментами *sdust* и *dustmasker*. В качестве тестовых данных использовался геном человека **GRCh38** (*GCF_000001405.40_GRC_h38.p14*) размером ~ 3,2 Гб.

Тестирование производительности проводилось на двух серверах:

e5: 2 × Intel Xeon CPU E5-2680 v4 @ 2.40GHz @ 2.40GHz (всего 28 ядер, 56 потоков); 188 Гб DDR4 оперативной памяти; GPU NVIDIA A4000 16 Гб; ОС Ubuntu 24.04 LTS; CUDA: 12.8.

gold: 2 × Intel Xeon Gold 6252 CPU @ 2.10GHz (всего 48 ядер, 96 потоков); 256 Гб DDR4 оперативной памяти; ОС Ubuntu 24.04 LTS.

Для оценки близости результатов определения регионов низкой сложности используется коэффициент сходства Жаккара. Он определяет меру сходства между двумя множествами A и B как отношение мощности их пересечения к мощности объединения:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \quad (7)$$

Значение $J(A,B) \in [0,1]$, где 0 соответствует полному различию множеств, а 1 – их полному совпадению.

3. Результаты

Проведен сравнительный анализ производительности *dustscan*, *sdust* и *dustmasker*. Каждый тест выполнен 20 раз, и рассчитаны медианные значения времени выполнения и стандартное отклонение. Результаты измерений представлены в таблице.

Время выполнения и параметры запуска для различных инструментов

Название	Инструмент	Время, мс	Конфигурация
SDUST	sdust	321 318 (± 664)	-w 64 -t 1
DustMasker	dustmasker	281 780 (± 3 784)	-w 64 -level 1
gpu-w64	dustscan	131 935 (± 85)	-w 64 -t 0,1 (без -O3)
gpu-o3-w64	dustscan	63 613 (± 432)	-w 64 -t 0,1
gpu-o3-w24	dustscan	32 909 (± 435)	-w 24 -t 0,1
e5-56-w64	dustscan	63 816 (± 878)	-w 64 -t 0,1 -c 56
e5-16-w64	dustscan	119 773 (± 9 504)	-w 64 -t 0,1 -c 16
e5-8-w64	dustscan	189 179 (± 3 385)	-w 64 -t 0,1 -c 8
e5-4-w64	dustscan	343 007 (± 3 366)	-w 64 -t 0,1 -c 4
e5-1-w64	dustscan	1 128 039 (± 14 927)	-w 64 -t 0,1 -c 1
gold-96-w64	dustscan	33 004 (± 543)	-w 64 -t 0,1 -c 96
gold-56-w64	dustscan	51 263 (± 645)	-w 64 -t 0,1 -c 56
gold-16-w64	dustscan	75 352 (± 1 059)	-w 64 -t 0,1 -c 16
gold-8-w64	dustscan	132 567 (± 1 754)	-w 64 -t 0,1 -c 8
gold-4-w64	dustscan	234 493 (± 3 103)	-w 64 -t 0,1 -c 4
gold-1-w64	dustscan	838 646 (± 11 098)	-w 64 -t 0,1 -c 1
e5-56-w24	dustscan	47 437 (± 716)	-w 24 -t 0,1 -c 56
e5-16-w24	dustscan	70 878 (± 994)	-w 24 -t 0,1 -c 16
e5-8-w24	dustscan	101 233 (± 1 404)	-w 24 -t 0,1 -c 8

Как видно из таблицы, наилучшие результаты продемонстрированы gpu-o3-w24 и gold-96-w24, что показывает ускорение в ~ 9,8 раз по сравнению с реализацией *sdust* и в ~ 8,6 раз по сравнению с *dustmasker*. Интересно, что CPU-режим *dustscan* с использованием 96 потоков среди рассматриваемых конфигураций показывает сопоставимую производительность с gpu-o3.

На основе полученных результатов рассчитаны параллельное ускорение и параллельная эффективность для CPU-реализации с различным количеством потоков. Параллельное ускорение определяется как отношение времени выполнения на одном потоке ко времени выполнения на N потоках:

$$S_N = \frac{T_1}{T_N} \tag{8}$$

Параллельная эффективность определяется как отношение параллельного ускорения к количеству потоков:

$$E_N = \frac{S_N}{N} = \frac{T_1}{N \times T_N} \tag{9}$$

На рис. 2 изображены графики параллельного ускорения (*a*) [21], параллельной эффективности (*b*) [21], масштабирования времени обработки (*c*). При сравнении времени выполнения оптимизированной сборки DUSTSCAN и неоптимизированной сборки (рис. 2, *d*) можно заметить разницу во времени выполнения анализа с и без применения флага компиляции «-O3», что может указывать на существенный размер накладных расходов CPU при чтении исходных данных, их передаче в видеопамять и обратно, а также записи в файл.

На рис. 3 продемонстрировано сравнение сторонних инструментов с нашим (в отобранных для сравнения конфигурациях).

Помимо производительности, произведена оценка качества обнаружения регионов низкой сложности при сравнении результатов DUSTSCAN с результатами SDUST с сопоставимыми параметрами с использованием в качестве оценки коэффициента сходства Жаккара (7). При сравнении множеств

регионов (с учетом их начальных и конечных координат) были получены пересечение в 2 939 517 111 нуклеотида (intersection), объединение в 3 136 906 988 нуклеотидов (union), метрика Жаккара 0,937075 при 39 961 409 перекрывающих сегментах (n_intersections).

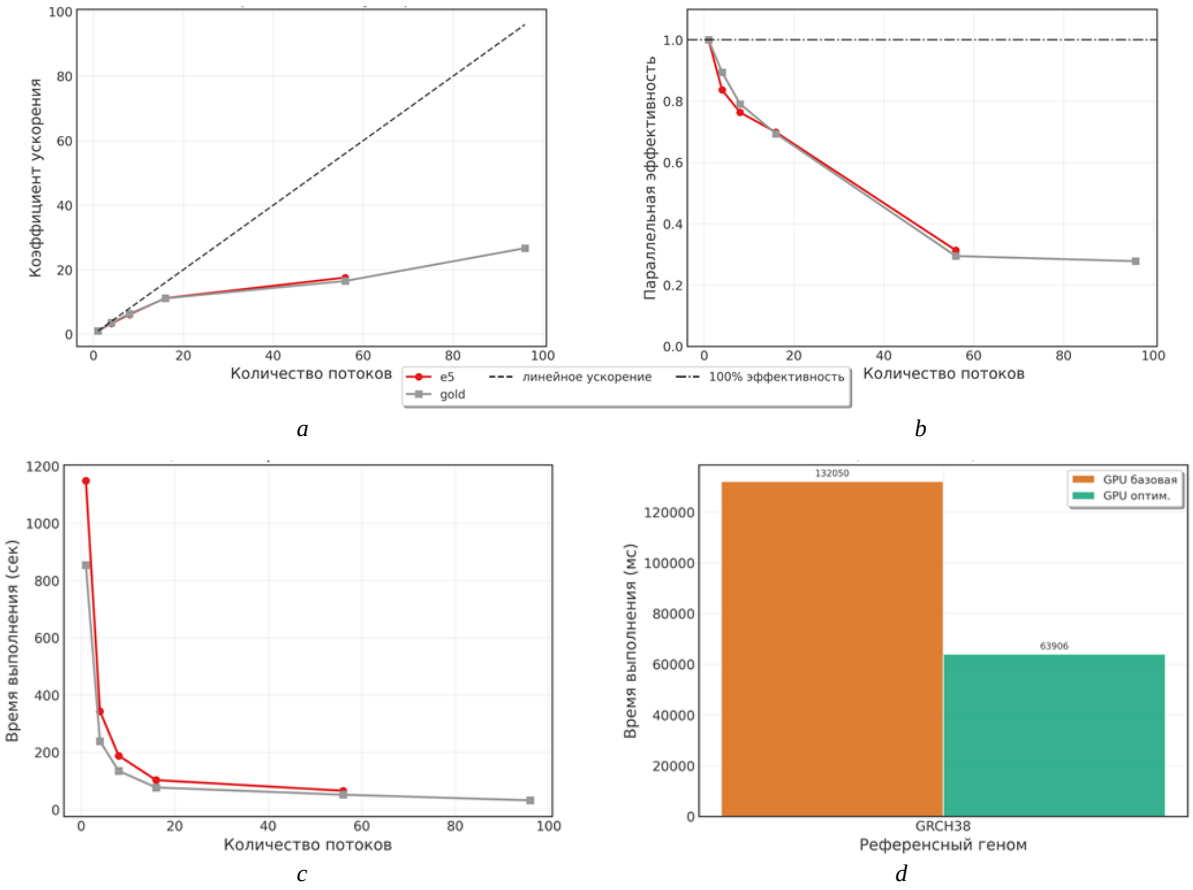

Рис. 2. Графики сравнения производительности: *a* – параллельное ускорение CPU (8); *b* – параллельная эффективность (9); *c* – масштабирование времени выполнения анализа; *d* – накладные CPU-расходы при использовании анализа с использованием GPU (неоптимизированная сборка и сборка, собранная с ключом «-O3»).

Fig. 2. Comparisons plots: *a*) CPU parallel acceleration (8); *b*) CPU parallel efficiency (9); *c*) scaling of analysis execution time; *d*) CPU overhead when using GPU analysis (non-optimized build and build compiled with the "-O3" flag).

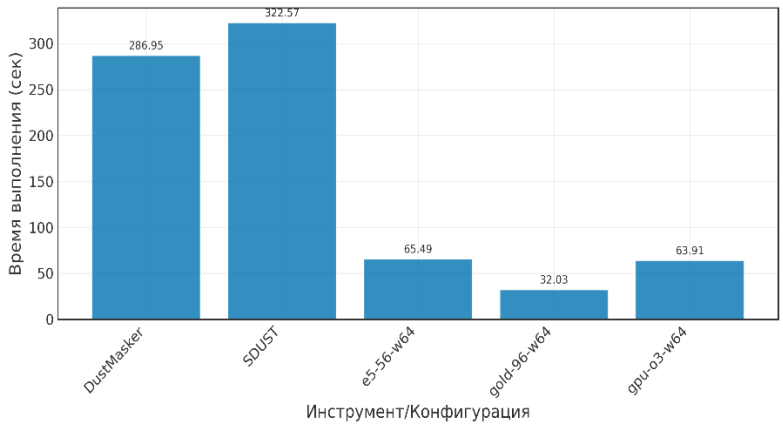

Рис. 3. Сравнение инструментов по времени выполнения анализа

Fig. 3. Comparison of tools by analysis time

Коэффициент сходства Жаккара ~ 0,94 указывает на высокое, но не идеальное совпадение между результатами DUSTSCAN и SDUST, однако стоит отметить, что DUSTSCAN гораздо менее

чувствителен к мелким регионам и успешно идентифицирует основные регионы низкой сложности, что может быть особенно важно для приложений, связанных с дизайном олигонуклеотидных ДНК-зондов.

Заключение

Проблема обнаружения LCR является чрезвычайно острой в рамках задачи подбора ДНК-зондов. На данный момент широко распространенные инструменты имеют ряд ограничений, которые призван разрешить алгоритм, предложенный в рамках данной работы.

DUSTSCAN представляет собой параллельную модификацию алгоритма DUST, что позволяет решать актуальную на данный момент задачу обнаружения LCR в геномных последовательностях.

Результаты показывают, что DUSTSCAN обеспечивает многократное ускорение (в 8–10 раз) по сравнению с SDUST при обработке полного генома человека, что существенно сокращает затрачиваемое на обнаружение LCR время. Ускорение относительно DustMasker составляет свыше 4 раз.

Интересно отметить, что CPU-режим *dustscan* с использованием 96 потоков показывает производительность, сопоставимую с GPU-режимом. Это можно объяснить затратами на передачу данных между CPU и GPU, а также накладными расходами на инициализацию и запуск CUDA-ядер [20].

Особенно эффективен DUSTSCAN при обработке очень длинных непрерывных геномных последовательностей.

Несмотря на значительное ускорение, DUSTSCAN имеет ряд ограничений. Он потребляет больше памяти, чем другие инструменты (за счет параллельного вычисления для различных регионов исходной последовательности). Размер обрабатываемого блока ограничен объемом доступной видеопамяти, что может быть проблемой для очень длинных последовательностей. Передача данных между CPU и GPU может создавать дополнительные накладные расходы, особенно для небольших последовательностей, где преимущества параллельной обработки не компенсируют эти расходы. Это же будет бутылочным горлышком при попытке реализации распределенной модификации алгоритма. Чем меньше размер последовательности, тем менее эффективным будет GPU-режим. Также DUSTSCAN может определять границы регионов иначе, чем SDUST, что может быть критично для некоторых приложений, особенно чувствительных к коротким последовательностям регионов низкой сложности.

В качестве направлений дальнейшего развития и улучшения DUSTSCAN можно назвать оптимизацию использования и работы с памятью и передачи данных между CPU и GPU, а также модификацию CUDA-ядер для выполнения полного цикла анализа (включая пост-обработку) на GPU.

В целом DUSTSCAN демонстрирует высокий потенциал использования распараллеливания для ускорения выделения из последовательности референсного генома регионов низкой сложности.

Список источников

1. Компо Ф., Певзнер. П. Алгоритмы биоинформатики. М. : ДМК-Пресс, 2023. 682 с.
2. Morgulis A., Gertz M., Schäffer A.A., Agarwala R. A Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences // Journal of Computational Biology. 2006. V. 13 (5). P. 1028–1040. doi: 10.1089/cmb.2006.13.1028
3. Orlov Y.L., Potapov V.N. Complexity: an internet resource for analysis of DNA sequence complexity // Nucleic Acids Research. 2004. V. 32. P. W628–W633. doi: 10.1093/nar/gkh466
4. Orlov Y.L., Orlova N.G. Bioinformatics tools for the sequence complexity estimates // Biophysical Reviews. 2023. V. 15. P. 1367–1378. doi: 10.1007/s12551-023-01140-y
5. Altschul S.F., Madden T.L., Schäffer A.A., Zhang J., Zhang Z., Miller W., Lipman D.J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs // Nucleic Acids Research. 1997. V. 25, is. 17. P. 3389–3402. doi: 10.1093/nar/25.17.3389
6. Goldfeder R.L., Priest J.R., Zook J.M., Grove M.E., Waggott D., Wheeler M.T., Salit M., Ashley E.A. Medical implications of technical accuracy in genome sequencing // Genome Medicine. 2016. V. 8. Art. 24. doi: 10.1186/s13073-016-0269-0
7. Koboldt D.C. Best practices for variant calling in clinical sequencing // Genome Medicine. 2020. V. 12. Art. 91. doi: 10.1186/s13073-020-00791-w
8. Lau T.Y. et al. The Neoantigen Landscape of the Coding and Noncoding Cancer Genome Space // The Journal of Molecular Diagnostic. 2022. V. 24 (6). P. 541–554. doi: 10.1016/j.jmoldx.2022.02.004

9. Shalon D., Smith S.J., Brown P.O. A DNA microarray system for analyzing complex DNA samples using two-color fluorescent probe hybridization // Genome research. 1996. V. 6 (7). P. 639–645. doi: 10.1101/gr.6.7.639
10. Haas B.J., Dobin A., Li B., Stransky N., Pochet N., Regev A. Accuracy assessment of fusion transcript detection via read-mapping and de novo fusion transcript assembly-based methods // Genome Biology. 2019. V. 20. Art. 213. doi: 10.1186/s13059-019-1842-9
11. Feng Y., Guo Q., Chen W., Han C. A Low-Complexity Deep Learning Model for Predicting Targeted Sequencing Depth from Probe Sequence // Applied Sciences. 2023. V. 13 (12). Art. 6996. doi: 10.3390/app13126996
12. Wootton J.C., Federhen S. Statistics of local complexity in amino acid sequences and sequence databases // Computers & Chemistry. 1993. V. 17, is. 2. P. 149–163. doi: 10.1016/0097-8485(93)85006-X
13. Velichkovski G., Gusev M., Mileski D. CUDA Calculation of Shannon Entropy for a Sliding Window System // 32nd Telecommunications Forum (TELFOR), November 2024. Belgrade : IEEE, 2024. P. 1–4. doi: 10.1109/TELFOR63250.2024.10819103
14. Frith M.C. A new repeat-masking method enables specific detection of homologous sequences // Nucleic Acids Research. 2011. V. 39, is. 4. Art. e23. doi: 10.1093/nar/gkq1212
15. Chen S., Zhou Y., Chen Y., Gu J. fastp: an ultra-fast all-in-one FASTQ preprocessor // Bioinformatics. 2018. V. 34, is. 17. P. i884–i890. doi: 10.1093/bioinformatics/bty560
16. Schmieder R., Edwards R. Fast Identification and Removal of Sequence Contamination from Genomic and Metagenomic Datasets // PLoS ONE. 2011. V. 6 (3). Art. e17288. doi: 10.1371/journal.pone.0017288
17. Kirk D.B., Hwu W.W. Programming Massively Parallel Processors: A Hands-on Approach, 3rd ed. Morgan Kaufmann, 2016. xix, 258 p.
18. Sanders J., Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley Professional, 2010. xix, 290 p.
19. Боресков А.В. и др. Параллельные вычисления на GPU. Архитектура и программная модель CUDA : учеб. пособие. М. : Изд-во Моск. ун-та, 2015. 336 с. (Суперкомпьютерное образование).
20. Jarnot P., Ziemska-Legiecka J., Grynberg M., Gruca A. Insights from analyses of low complexity regions with canonical methods for protein sequence comparison // Briefings in Bioinformatics. 2022. V. 23, is. 5. Art. bbac299. doi: 10.1093/bib/bbac299
21. Замятин А.В. Интеллектуальный анализ данных: учебное пособие. Томск : Изд. Дом Том. гос. ун-та, 2020. 196 с.

References

1. Compeau, P. & Pevzner, P. (2018) *Algoritmy bioinformatiki* [Bioinformatics Algorithms]. Moscow: DMK-press.
2. Morgulis, A., Gertz, M., Schäffer, A.A. & Agarwala, R. (2006) A Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences. *Journal of Computational Biology*. 13(5). pp. 1028–1040. DOI: 10.1089/cmb.2006.13.1028
3. Orlov, Y.L. & Potapov, V.N. (2004) Complexity: an internet resource for analysis of DNA sequence complexity. *Nucleic Acids Research*. 32. pp. W628–W633. DOI: 10.1093/nar/gkh466
4. Orlov, Y.L. & Orlova, N.G. (2023) Bioinformatics tools for the sequence complexity estimates. *Biophysical Reviews*. 15. pp. 1367–1378. DOI: 10.1007/s12551-023-01140-y
5. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W. & Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*. 25(17). pp. 3389–3402. DOI: 10.1093/nar/25.17.3389
6. Goldfeder, R.L., Priest, J.R., Zook, J.M., Grove, M.E., Waggott, D., Wheeler, M.T., Salit, M. & Ashley, E.A. (2016) Medical implications of technical accuracy in genome sequencing. *Genome Medicine*. 8. Art. 24. DOI: 10.1186/s13073-016-0269-0
7. Koboldt, D.C. (2020) Best practices for variant calling in clinical sequencing. *Genome Medicine*. 12. Art. 91. DOI: 10.1186/s13073-020-00791-w
8. Lau, T.Y. et al. (2022) The Neoantigen Landscape of the Coding and Noncoding Cancer Genome Space. *The Journal of Molecular Diagnostic*. 24(6). pp. 541–554. DOI: 10.1016/j.jmoldx.2022.02.004
9. Shalon, D., Smith, S.J. & Brown, P.O. (1996) A DNA microarray system for analyzing complex DNA samples using two-color fluorescent probe hybridization. *Genome Research*. 6(7). pp. 639–645. DOI: 10.1101/gr.6.7.639
10. Haas, B.J., Dobin, A., Li, B., Stransky, N., Pochet, N. & Regev, A. (2019) Accuracy assessment of fusion transcript detection via read-mapping and de novo fusion transcript assembly-based methods. *Genome Biology*. 20. Art. 213. DOI: 10.1186/s13059-019-1842-9
11. Feng, Y., Guo, Q., Chen, W. & Han, C. (2023) A Low-Complexity Deep Learning Model for Predicting Targeted Sequencing Depth from Probe Sequence. *Applied Sciences*. 13(12). Art. 6996. DOI: 10.3390/app13126996
12. Wootton, J.C. & Federhen, S. (1993) Statistics of local complexity in amino acid sequences and sequence databases. *Computers & Chemistry*. 17(2). pp. 149–163. DOI: 10.1016/0097-8485(93)85006-X
13. Velichkovski, G., Gusev, M. & Mileski, D. (2024) CUDA Calculation of Shannon Entropy for a Sliding Window System. *Proceedings IEEE, 32nd Telecommunications Forum (TELFOR).* November 2024. DOI: 10.1109/TELFOR63250.2024.10819103
14. Frith, M.C. (2011) A new repeat-masking method enables specific detection of homologous sequences. *Nucleic Acids Research*. 39(4). Art. e23. DOI: 10.1093/nar/gkq1212
15. Chen, S., Zhou, Y., Chen, Y. & Gu, J. (2018) fastp: an ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics*. 34(17). pp. i884–i890. DOI: 10.1093/bioinformatics/bty560
16. Schmieder, R. & Edwards, R. (2011) Fast identification and removal of sequence contamination from genomic and metagenomic datasets. *PLoS ONE*. 6(3). Art. e17288. DOI: 10.1371/journal.pone.0017288

17. Kirk, D.B. & Hwu, W.W. (2016) *Programming massively parallel processors*: *A Hands-on Approach*. 3rd ed. Morgan Kaufmann.
18. Sanders, J. & Kandrot, E. (2010) *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional.
19. Boreskov, A.V. & et al. (2015) *Parallel'nye vychisleniya na GPU. Arkhitektura i programmnaya model' CUDA* [Parallel computing on GPU. CUDA architecture and programming model]. Moscow: Moscow University.
20. Jarnot, P., Ziemska-Legiecka, J., Grynberg, M. & Gruca, A. (2022) Insights from analyses of low complexity regions with canonical methods for protein sequence comparison. *Briefings in Bioinformatics*. 23(5). Art. bbac299. DOI: 10.1093/bib/bbac299
21. Zamyatin, A.V. (2020) *Intellektual'nyy analiz dannykh* [Data Mining]. Tomsk: Tomsk State University.

***Информация об авторах*:**
Воробьёв Ростислав Сергеевич – младший научный сотрудник Научно-исследовательского института онкологии Томского национального исследовательского медицинского центра Российской академии наук (Томск, Россия); аспирант Института прикладной математики и компьютерных наук Национального исследовательского Томского государственного университета (Томск, Россия). E-mail: tsu@rvorobev.ru
Замятин Александр Владимирович – профессор, доктор технических наук, директор Института прикладной математики и компьютерных наук Национального исследовательского Томского государственного университета (Томск, Россия). E-mail: zamyatin@mail.tsu.ru
Геращенко Татьяна Сергеевна – кандидат медицинских наук, старший научный сотрудник Научно-исследовательского института онкологии Томского национального исследовательского медицинского центра Российской академии наук (Томск, Россия). E-mail: t_gerashchenko@oncology.tomsk.ru
Коробейникова Анастасия Алексеевна – младший научный сотрудник Научно-исследовательского института онкологии Томского национального исследовательского медицинского центра Российской академии наук (Томск, Россия). E-mail: shegolmay@gmail.com
Денисов Евгений Владимирович – доктор биологических наук, заведующий лабораторией Научно-исследовательского института онкологии Томского национального исследовательского медицинского центра Российской академии наук (Томск, Россия). E-mail: d_evgeniy@oncology.tomsk.ru

Вклад авторов: все авторы сделали эквивалентный вклад в подготовку публикации. Авторы заявляют об отсутствии конфликта интересов.

***Information about the authors*:**
Vorobev Rostislav S. (Junior Researcher, Cancer Research Institute, Tomsk National Research Medical Center, Russian Academy of Sciences, Tomsk, Russian Federation; Post-Graduate Student, National Research Tomsk State University, Tomsk, Russian Federation). E-mail: tsu@rvorobev.ru
Zamyatin Alexander V. (Doctor of Technical Sciences, Professor, National Research Tomsk State University, Tomsk, Russian Federation). E-mail: zamyatin@mail.tsu.ru
Gerashchenko Tatiana S. (Candidate of Medical Sciences, Cancer Research Institute, Tomsk National Research Medical Center, Russian Academy of Sciences, Tomsk, Russian Federation). E-mail: t_gerashchenko@oncology.tomsk.ru
Korobeynikova Anastasia A. (Junior Researcher, Cancer Research Institute, Tomsk National Research Medical Center, Russian Academy of Sciences, Tomsk, Russian Federation). E-mail: shegolmay@gmail.com
Denisov Evgeny V. (Doctor of Biological Sciences, Cancer Research Institute, Tomsk National Research Medical Center, Russian Academy of Sciences, Tomsk, Russian Federation. E-mail: d_evgeniy@oncology.tomsk.ru

Contribution of the authors: the authors contributed equally to this article. The authors declare no conflicts of interests.

Поступила в редакцию 10.06.2025; принята к публикации 02.09.2025

Received 10.06.2025; accepted for publication 02.09.2025