ELSEVIER

From GPUs to AI and quantum: three waves of acceleration in bioinformatics

Bertil Schmidt *, **Andreas Hildebrandt** *

Institut für Informatik, Johannes Gutenberg University, Mainz, Germany

The enormous growth in the amount of data generated by the life sciences is continuously shifting the field from model-driven science towards data-driven science. The need for efficient processing has led to the adoption of massively parallel accelerators such as graphics processing units (GPUs). Consequently, the development of bioinformatics methods nowadays often heavily depends on the effective use of these powerful technologies. Furthermore, progress in computational techniques and architectures continues to be highly dynamic, involving novel deep neural network models and artificial intelligence (AI) accelerators, and potentially quantum processing units in the future. These are expected to be disruptive for the life sciences as a whole and for drug discovery in particular. Here, we identify three waves of acceleration and their applications in a bioinformatics context: (i) GPU computing, (ii) AI and (iii) next-generation quantum computers.

BertilSchmidt is tenured full professor of high-performance computing at Johannes Gutenberg University Mainz and an adjunct professor at SIT (Singapore Institute of Technology). Prior to that, he was a faculty member at NTU (Nanyang Technological University). His research group has designed and implemented a variety of massively parallel algorithms and tools focusing on the analysis of large-scale bioinformatics data sets on numerous platforms including graphics processing units (GPUs), field-programmable gate arrays and various supercomputers. For his pioneering research work on GPU computing, he has received one of the first CUDA Academic Partnership awards, a CUDA Professor Partnership and three IEEE Best Paper Awards (IEEE ASAP 2009, IEEE ASAP 2015 and IEEE HiPC 2020). His active collaboration with Shandong University has led to various parallel methods for life-science applications that can scale towards tens of thousands of nodes on world-leading supercomputers. He serves as associate editor of the *Journal of Parallel and Distributed Computing* and the *Journal of Computational Science*. He also authored the textbook *Parallel Programming: Concepts and Practice*, which provides an upper-level introduction to parallel programming.

Andreas Hildebrandtis tenured full professor and chair for bioinformatics and software engineering at Johannes Gutenberg University Mainz in Germany. Previously, he was the head of the independent junior research group on protein–protein interactions and computational proteomics at the Center for Bioinformatics at Saarland University in Saarbrücken, Germany. His main research interests include structural bioinformatics, computational proteomics and computational systems biology, and his group develops and applies novel techniques and program packages. He is also one of the core developers of the BALL (Biochemical Algorithms Library) library.

Introduction

Over the past decade we have witnessed a tremendous increase in the volume of data generated in the life sciences, propelled in particular by the rapid progress of high-throughput technologies such as sequencing machines, imaging technologies and mass spectrometers. It has been estimated that between 100 million and 2 billion human genomes could be sequenced by 2025,[p1] and this explosion of sequencing data is already having a major impact on the fields of personalised medicine and drug discovery, as well as on many areas of life in general.[p2] Analysing the massive data sets produced by modern high-throughput instruments, however, poses difficult computational challenges,[p3] which has led to the use of modern high-performance computing (HPC) infrastructures in the life sciences.

The accelerator approach in computer architecture is based on specialised processors connected to a host central processing unit (CPU) that offloads suitable computational kernels (see Figure 1). The choice of a particular accelerator technology usually depends on their specific strengths and various other considerations. For example, graphics processing units (GPUs) offer broad applicability and cost-effectiveness, making them a popular choice for a

* *Corresponding authors. Schmidt, B. (bertil.schmidt@uni-mainz.de), Hildebrandt, A. (andreas.hildebrandt@uni-mainz.de).*

1359-6446/
https://doi.org/10.1016/j.drudis.2024.103990

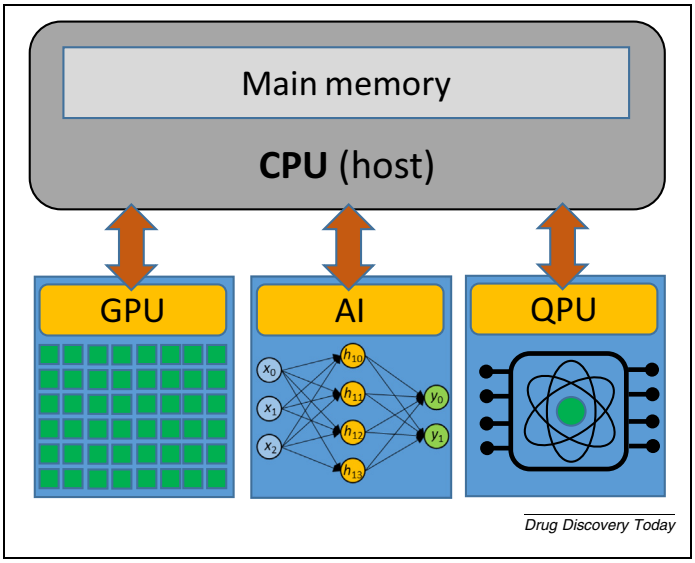


FIGURE 1
The accelerator model of computing. Specialised processing units are connected to a generic host CPU that offloads compute-intensive tasks to enable faster execution. Different accelerator technologies, such as GPUs, AI accelerators or QPUs, can be chosen based on their strengths.

variety of bioinformatics tasks, whereas field-programmable gate arrays (FPGAs) provide a highly customizable and power-efficient solution for specific applications such as Illumina's DRAGEN Bio-IT platform for variant calling.[p4] FPGA configurations are generally specified with a hardware description language, which can make them difficult to program and challenging to port to other machines. In addition, there is a lack of flexibility to modify the hardware implementation of the synthesised code to different algorithmic variants.

A key enabler of the first wave of acceleration was therefore the release and widespread adoption of the Compute Unified Device Architecture (CUDA) programming language for Nvidia GPUs in 2007, which opened the era of general purpose computation on GPUs (GPGPU).[p5] Typical first-wave applications in the life sciences include the acceleration of database searches, read mapping, variant calling, *de novo* assembly, molecular dynamics simulations and molecular docking.

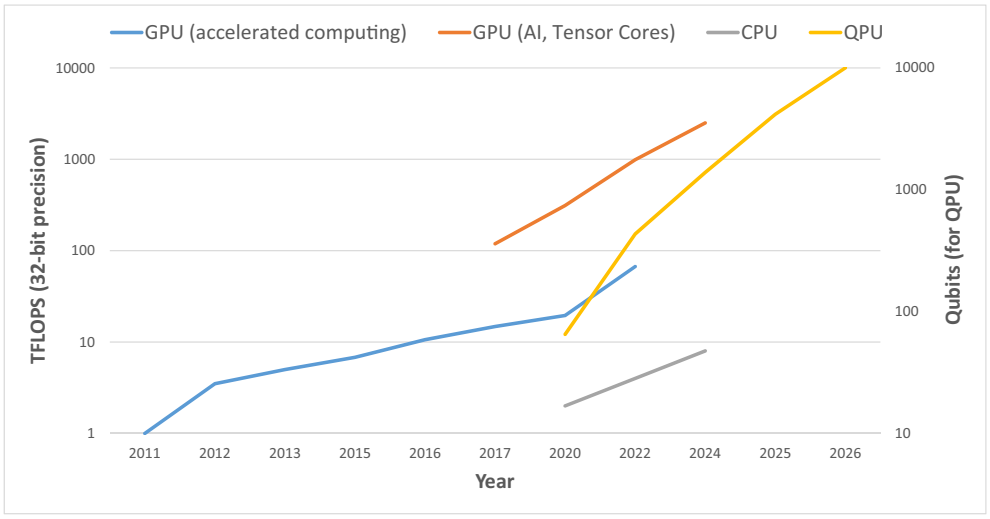
The second wave of acceleration was initiated by the release of the CUDA Deep Neural Network (cuDNN) library in 2014 for speeding up training and inference computation for deep neural networks (DNNs) on GPUs.[p6] This laid the foundation for the revolutionary adoption of methods from statistical learning that are not'seeded' with a biological model in mind, but rather attempt to learn directly from the data. Artificial neural networks (ANNs) have a long history in machine learning (ML), reaching back to Rosenblatt's biologically inspired Perceptron model[p7] from 1958. Early ANNs were severely restricted in their size (the number of layers and number of neurons per layer), and consequently they were unable to compete with other popular ML techniques on complex tasks. Groundbreaking improvements in computer hardware, neural network design and algorithmic approaches for the training of ANNs, however, led to the development of so-called deep learning (DL) methods, in which ANNs typically contain large numbers of hidden layers. Such deep networks have allowed revolutionary results in fields such as computer vision and voice or text recognition. Consequently, we have seen the widespread adoption of a number of cutting-edge DL techniques, such as transformers and large language models, for biomedical data during the second wave. These methods are currently revolutionising the life sciences, and they are enabled by modern artificial intelligence (AI) accelerator technologies and the development of corresponding DL libraries.[p8] However, increasingly complex neural networks and data sets require even higher performance, which has motivated the development of improved AI accelerators such as Nvidia's Tensor Cores (TCs) or Google's Tensor Processing Units (TPUs).

Quantum computing (QC) is an alternative computing paradigm based on quantum mechanics. Quantum computers can in principle solve certain problems much faster than their classical counterparts. This technology is still emerging, and we have currently not reached the advent of useful quantum computation; however, there are already a few early quantum accelerator systems available as co-processors, which have demonstrated their suitability for selected problems in sequence-based and structure-based bioinformatics.[p9],[p10] This shows the promise of QC as the third wave of acceleration in the future.

Figure 2 summarises the development of processing power for the considered three waves of acceleration. It shows the rapid development of the compute power of Nvidia GPUs in terms of peak performance in teraFLOPS (TFLOPS) using CUDA cores from 2011 to 2022 and TCs for AI from 2017 to 2024. In addition, the expected progress of quantum processors in terms of the number of quantum bits (qubits) is shown according to the IBM quantum roadmap.

Compute clusters built around powerful multicore CPU servers traditionally play an important part in accelerating bioinformatics pipelines. Whereas CPUs can process many general tasks in a fast, sequential manner, GPUs use massive parallelism to break down problems into a large number of smaller, simultaneous calculations. We show in the following sections that a variety of algorithms for bioinformatics and AI can be efficiently mapped onto this type of architecture. As shown in Figure 2, GPUs can offer between one and two orders-of-magnitude higher peak performance. For example, the suffix array of a full human genome sequence was recently constructed on a single GPU-based DGX server within 3.4 seconds,[p11] whereas it previously took 4.8 seconds when using 100 nodes on a CPU-based compute cluster. This shows that efficient usage of accelerator technology can offer more effective solutions in terms of both cost and energy consumption.

Consequently, most HPC systems are nowadays equipped with accelerators such as GPUs. They are important for sustaining Moore's law, which predicts that processing powers doubles roughly every two years, and they are crucial for numerous workloads such as those using large language models and quantum mechanics simulations in drug discovery pipelines. We highlight accelerated methods in bioinformatics within the identified three waves: (i) GPGPU, (ii) AI and (iii) QC, in connection with corresponding enabling technologies. Bioinformatics is a wide field. We therefore focus on sequence-based and structure-based applications. Further popular areas such as systems biology



Drug Discovery Today

FIGURE 2

The development of processing performance for the three waves of acceleration. (i) First wave: the GPU (accelerated computing) line (blue) shows the peak performance in TFLOPS (single precision) of Tesla GPUs (C2050, K20, K40, M40, P100, V100, A100 and H100). (ii) Second wave: the GPU (AI, tensor cores) line (orange) shows the peak performance in TFLOPS for Tensor Cores in TF32 format using the sparsity feature (if available) on V100, A100, H100 and B200 GPUs. (iii) Third wave: the QPU line (yellow) shows the planned development of the number of qubits on IBM quantum processors. TFLOPS peak performance (single precision) for typical high-end multi-core server CPUs is also shown as reference (grey line).

or synthetic biology are outside the scope of our review. Our reviewed bioinformatics methods are summarised in Table 1 and highlighted in the following sections.

First wave: GPGPU

Enabling technologies

Progress in computer architecture has recently focused on the emergence of accelerators to which the host CPU can offload suitable computational tasks. Relevant accelerator technologies come in different variants, each with their own characteristics and strengths. In the following three sections, we briefly discuss three popular types: GPUs, AI accelerators and quantum processing units (QPUs).

GPGPU refers to the use of GPUs to perform computations in applications outside computer graphics. CUDA was released in 2007 as an API that allows the use of C/C++ or Fortran to implement massively parallel algorithms for execution on Nvidia GPUs. This started the widespread adoption of GPUs in many scientific disciplines. GPUs can provide around one order-of-magnitude higher peak performance when compared with CPUs through massive fine-grained parallelism executed on more than 10,000 cores on a single chip. In addition to higher computational power, GPUs feature high-bandwidth memories (HBMs) with throughput of up to 8 TB/s for systems with HBM3e. Although the number of cores and memory bandwidth of GPUs is much higher than that of contemporary CPUs, their main memory capacity is usually smaller. To unlock the power of GPUs, careful design of algorithms and data structures that can efficiently exploit fine-grained parallelism is therefore required.[p12]

By using programming frameworks such as CUDA, OpenCL or AMD's ROCm, GPUs can be used for general-purpose applications. Implementing hand-optimised algorithms from scratch using programming languages such as CUDA can be time-consuming because it requires deep understanding of GPU architectures. However, the advancements of pragma-based languages such as OpenACC, scientific computing libraries and cross-platform abstraction layers such as SYCL can facilitate higher level yet efficient programming.

Sequence-based bioinformatics

BLAST. The most popular software for searching a library of sequences with a query sequence is BLAST. BLAST is a family of programs, of which BLASTN (Nucleotide BLAST) and BLASTP (Protein BLAST) are the most widely used. The heuristics used consist of a pipeline of several stages, which have different characteristics but also contain common components. CUDA-BLASTP[p13] and H-BLAST[p14] are implementations of BLASTP on CUDA-enabled GPUs with speed increases ranging between 4 and 10 on one K20x GPU over the sequential NCBI-BLASTP. G-BLASTN[p15] also reports massively parallel GPU acceleration of BLASTN.

Read mapping and variant calling. The mapping of next-generation sequencing (NGS) reads to a reference genome sequence is an important stage in data-processing pipelines. Many existing read mappers are based on a seed-and-extend approach, in which short exact matches are first identified using an index data structure. These seeds are then verified to see whether they can be extended to a full alignment. Approaches for accelerating read mapping can be categorised by the indexing data structure they use. Early GPU approaches, including CUSHAW[p16] and SOAP3-dp[p17] were based on accelerating

TABLE 1

Summary of reviewed methods for the three waves in selected application areas

Wave	Application	Approach	Bioinformatics methods
GPGPU	BLAST	DB search	CUDA-BLASTP,[p13] H-BLAST,[p14] G-BLASTN[p15]
	Read mapping	BWT	CUSHAW,[p16] SOAP3-dp,[p17] BWA-MEM2[p20]
		Hash table	Arioc,[p18] MetaCache,[p19] mm2-ax[p21]
	Variant calling	Pipeline	BALSA,[p22] ParaBricks[p23]
	De novo assembly	Graphs	LaSAGNA,[p24] MetaHipMer[p25]
	Error correction	Hashing	DecGPU,[p26] CARE[p27]
	Pairwise alignment	DP	CUDASW++,[p35] CUDAlign,[p31] Adept,[p33] GASAL2,[p32] AnySeq,[p34] WFA-GPU[p36]
	RNA folding	DP	Nussinov[p41],[p44] RNAfold,[p43] Partition[p42]
	MSA	PA	MSA-CUDA,[p37] G-MSA,[p38] QuickProbs[p39]
	MD	Simulation	AMBER,[p45] GROMACS,[p46] NAMD[p47]
	Protein docking	Optimisation	ClusPro,[p50] MEGADOCK,[p51] Uni-Dock,[p57] AutoDock4-GPU,[p53] Vina-GPU,[p56] hex[p52]
AI	Variant calling	CNN	DeepVariant,[p62] PEPPER,[p63] ClairS[p64]
	Error correction	Transformer	DeepConsensus[p65]
	Annotation	Transformer	DNABERT,[p66],[p67] Enformer,[p68] NT[p69]
	Metagenomics	LSTM	DeepMicrobes[p70]
		Transformer	MetaTransformer[p71]
	Base calling	Various	Chiron,[p73] Bonito,[p74] SAcall,[p75] RUBICON[p76]
	Protein folding	Transformer	AlphaFold,[p77],[p106] ESMFold,[p82] trRosettaX[p83]
	Protein design	Transformer	ESM[p84],[p85]
		Diffusion	FrameDiff,[p88] ProteinDiffusionGenerator[p89]
	MD	CNN	SchNet,[p90] TorchMD[p91]
		LSTM	LSTM-predict-MD[p93],[p94]
	Protein docking	CNN/GNN	DeepSite,[p95] GNINA,[p99] EquiBind[p100]
		Diffusion	DiffDock[p102]
QC	*De novo* assembly	Hamiltonian	QuASeR,[p110] QUBO[p109]
	Read mapping	Grover search	QiBAM[p111]
	MSA	Hamiltonian	MSA-QAOA[p112]
	RNA folding	Hamiltonian	QBM[p113]
	Computational chemistry	Hamiltonian	VQE,[p116],[p117] QPE[p10]

exact matching with the Burrows-Wheeler transform (BWT) and FM-index, and featured a low memory footprint that fitted into the limited GPU global memory. Arioc[p18] and MetaCache-GPU[p19] are more recent examples based on (subsampled) hash tables that also support modern multi-GPU systems. Even newer approaches provide GPU acceleration of short read mapping with BWA-MEM2[p20] and of long read mapping with Minimap2.[p21] Variant calling is an important operation performed on the alignments returned by mapping NGS reads to reference genomes.

Luo *et al.*[p22] leveraged the power of GPUs to implement a complete pipeline called BALSA that consists of read mapping, re-alignment and variant calling. ParaBricks[p23] is a state-of-the-art GPU-accelerated software suite for accelerating genomic workflows. It can achieve up to 65 × acceleration with germline variant callers, bringing Genome Analysis Toolkit (GATK) HaplotypeCaller runtimes down from 36 hours to 24 minutes using a system with eight A100 GPUs.

De novo *assembly and error correction*. Approaches for accelerating *de novo* genome assembly have been limited because assembly workloads are highly data-intensive, leading to large memory requirements for associated graph data structures. LaSAGNA[p24] uses a streaming approach that makes a number of passes over the input data based on the available GPU memory to build a string overlap graph. This leads to a runtime of 5 hours on an eight-node cluster of K20 GPUs for assembling a 400 GB human genome data set. MetaHipMer[p25] is a large-scale metagenome assembler for performing local assemblies based on de Bruijn graphs for metagenomic data on GPU clusters.

The time-consuming preprocessing step for correcting errors in NGS reads has also received considerable attention. DecGPU[p26] speeds up error correction on GPU clusters based on Bloom filters. CARE[p27],[p28] is a new method that corrects errors in Illumina reads based on computing multiple sequence alignments (MSAs) using accelerated hash tables, and it achieves order-of-magnitude lower false positive rates than state-of-the-art tools while being efficient when executed on GPUs.

Pairwise and multiple sequence alignment. For pairwise sequence similarity computation, the Smith-Waterman (SW) algorithm and the Needleman Wunsch (NW) algorithm, along with their variants, are widely used. They compute the optimal local, global or semi-global alignment of two sequences under a given scoring scheme by means of dynamic programming (DP). Performance across different accelerator platforms can be compared in terms of billion cell updates per second (GCUPS).

Manavski and Valle,[p29] who achieved a performance of up to 3.6 GCUPS on a GeForce 8800 GTX, proposed the first implementation of SW using CUDA. CUDASW++3.0[p30] considerably improved performance for a SW-based protein sequence database search to 185.6 GCUPS on a dual-GPU GeForce GTX 690. CUDAlign focuses on the computation of a single pairwise alignment of long (chromosome-length) DNA sequences. The latest version[p31] achieves 10,370 GCUPS on a cluster with 384 GPUs for computing exact chromosome-wide DNA alignments.

Current work focuses on the design of libraries for various types of pairwise alignments on different accelerators, such as GASAL2[p32] and ADEPT.[p33] The AnySeq/GPU[p34] library achieves over 80% peak performance on modern GPUs and outperforms GASAL2 and ADEPT by a median speed increase of 30 × when running on the same GPU hardware. The most recent CUDASW++4.0[p35] can exploit novel DPX instructions to accelerate DP on Hopper GPUs, and it can achieve up to 5,700 GCUPS for an SW-based protein sequence database search on a single H100 GPU or more than 36,000 GCUPS on a server with eight H100 GPUs. WFA-GPU[p36] accelerates semi-global pairwise alignments of short and long reads based on the wavefront algorithm (WFA) for different sequencing technologies on GPUs.

Progressive alignment is a widely used but time-consuming approach for computing MSAs. MSA-CUDA[p37] was the first GPU-implementation of all three stages of the ClustalW pipeline. G-MSA[p38] accelerates the T-Coffee algorithm to achieve a speed increase of around two orders-of-magnitude. QuickProbs[p39] is a variant of the highly accurate MSAProbs[p40] algorithm suited for GPUs.

RNA folding. RNA folding algorithms attempt to predict the secondary structure from a given RNA sequence. Corresponding algorithms are often based on DP and exhibit high computational complexities ranging from $O(n^3)$ to $O(n^6)$. Li *et al.*[p41] developed an accelerated version of Nussinov's algorithm achieving an order-of-magnitude speed increase compared with a multi-core CPU version. Stern and Mathews[p42] accelerated the computation of RNA secondary structure partition functions on GPUs, allowing the calculation of base-pair probabilities for large sequences in a reasonable amount of time, with a negligible compromise in accuracy due to working in single precision. Langdon and Lorenz[p43] integrated GPU acceleration into RNAfold so that many thousands of RNA secondary structures can be computed in parallel. More recent work[p44] further enhances the efficiency of Nussinov's algorithm on GPUs by integrating tiled and sparsified parallel Four-Russians techniques.

Structure-based bioinformatics

Whereas sequence-based bioinformatics generates biological insights from the properties of the sequences of DNA, RNA and proteins, structure-based bioinformatics analyses 3D representations of molecules and their interactions, and as well as the four-dimensional trajectories of molecular coordinates, interactions and derived properties as they change over time.

Molecular dynamics. Some of the most popular and powerful computational tools for studying molecular properties from a structural representation are molecular dynamics (MD) simulations. MD starts with specifications of the molecular topology, its initial configurations (the coordinates and velocities of all atoms) and a molecular force field (i.e., a mathematical specification of all pairwise interaction types in the system, together with a set of parameters for these interactions for each interacting pair of atoms). To infer molecular trajectories, MD software solves Newton's equations of motion numerically in iterative fashion and employs further algorithms (e.g., thermostats, barostats, etc.) to achieve constant temperature or pressure. Numerical stability dictates that the time step between subsequent iterations of the solver is small, often on the order of femtoseconds ($1e-15s$). The biological processes studied by MD simulations, however, typically live on much larger time scales, at least on the order of nanoseconds or microseconds, leading to millions or even billions of iterations. A further computational challenge is the quadratic growth of potential interactions with the number of atoms in the system. When including water molecules and the required counter ions, simulating a single protein often requires handling hundreds of thousands of atoms. This can easily lead to billions of potential interactions. Using clever algorithms and numerical approximation techniques, the number of interactions computed in practice can often be considerably reduced, but the computational effort is still tremendous. Fortunately, computing these interactions maps very well to the capabilities of GPUs, which have consequently been quickly adopted for the purpose of MD simulations.[p45]

The performance breakthrough achieved by GPU-based acceleration has been transformative: computations that used to require significant time and resources on supercomputers can now be routinely performed on workstations. MD software packages, such as GROMACS[p46] or NAMD,[p47] have seen continuous improvements whereby more and more computation has been shifted to the GPU, and they have adopted algorithms more suited to their compute architectures. The GPU technology available at the time of writing allows for generating hundreds of nanoseconds for medium-sized proteins, and even 1–10 ns for systems with millions of atoms, on a single machine with a single, albeit powerful, GPU.

Protein docking. Protein docking refers to the task of predicting the structure and binding free energy of a protein in complex with one or several other molecules. Currently, most docking algorithms focus on binary complexes, even though some work has been performed to allow multimeric complex assembly.[p48] The two main challenges in docking are the quality of the scoring of candidate complexes, and the treatment of molecular flexibility. For these reasons, protein–protein docking is often considered a simpler task than protein–ligand docking, even though proteins are considerably larger than ligand molecules (which are often drug candidates). The chemical regularity of proteins as polymers formed from amino acids often allows the reduction of flexibility to a discrete set of side chain rotamers, and potentially backbone torsion angle variation, thus greatly simplifying energetic computation because the types of atoms and their interactions are restricted.

But even with these simplifications, computing energy-based scoring functions, and handling the large degrees of freedom arising from flexibility, is still a challenge. Although many of the approaches used in protein docking share certain ideas with MD simulations, the computational techniques involved are related to numerical optimisation rather than differential equation solvers. The scoring schemes and optimisation approaches do not always translate as readily to GPU architectures as those involved in MD simulations, but still, many approaches can be significantly accelerated. In protein–protein docking, popular approaches base their optimisation on a rigid-body-docking idea initially presented in ref.[p49] Here, the proteins are represented on discrete 3D grids, crafted in such a way that their (discrete) correlation grid is closely related to their geometric fit. Energetic components can be included by, for example, extending the

scoring scheme to complex numbers. Computational bottlenecks in these approaches are the calculation of electrostatic fields and the computation of the correlation grids, which is based on the fast Fourier transform.

Both of these steps can profit from vectorisation, and most modern protein–protein docking software (such as ClusPro,[p50] MEGADOCK 4.0[p51] or hex[p52]) allows for offloading to GPUs. In protein–ligand docking, achievable GPU speed increases vary between different programs and the details of the system under consideration. AutoDock4-GPU,[p53] for instance, reports speed increases of between 2× and 350× compared with the single-threaded AutoDock4 implementation. Whereas AutoDock4 uses a Lamarckian genetic algorithm to optimise a relatively complex semi-empirical approximation of the binding free energy, the related AutoDock Vina suite uses a simplified scoring scheme and replaces the genetic algorithm with an iterative local search. Both changes help to improve computational performance and scalability, without significant deterioration of docking accuracy (in fact, the accuracy often even improves in practice). Two popular acceleration schemes, QuickVina 2[p54] and QuickVina-W,[p55] further reduce running times, but at the cost of degraded performance. The Vina-GPU 2.0[p56] package provides GPU-based implementations of all three schemes — AutoDock Vina, QuickVina 2 and QuickVina-W — reporting speed increases of 65.6×, 1.4× and 3.6×, respectively. Another AutoDock Vina-like approach, Uni-Dock,[p57] also reports enormous speed increases (compared with the single-core version of AutoDock Vina), which scale linearly with the number of GPUs used.

Second wave: AI

Enabling technologies

AI accelerators. AI accelerators are class of processors that are designed to accelerate neural network arithmetic operations on the basis of matrix multiplication. They contain engines that improve the performance of DL training and inference by accelerating corresponding linear algebra computations. Prominent examples include TCs on Nvidia GPUs, Google's TPUs or AMD's Matrix Cores. They are designed to perform multiplication of small matrices or tensors (e.g., of 4×4) at very high speed.[p58] Handling of bigger matrices is performed by partitioning into small sub-blocks, block multiplication and aggregation. They typically also feature fast support for a wide range of low-precision arithmetic to achieve even higher speeds, such as TensorFloat-32 (TF32), floating-point 16 (FP16) or FP8 formats. For example, a single H100 GPU can deliver a peak performance of around 1, 2 or 4 petaFLOPS when using TCs with TF32, FP16 or FP8 precision, whereas its peak performance without TCs is only 34 or 67 TFLOPS when using FP64 or FP32, respectively (see Figure 2).

Multiple accelerators are often used to further speed up the training of large-scale models. Approaches can be classified into model and data parallelism.[p59] The latter requires that each accelerator holds a copy of the complete neural network in its local memory. Synchronous training then sends different mini-batches of the training data to each accelerator for training. Modern multi-GPU systems feature fast interconnects to support expensive all-reduce communication operations, which are required to collect all trainable parameters between GPUs. For example, Blackwell-based DGX servers with eight GPUs contain a fifth-generation NVLink interconnect that provides 1,800 GB/s GPU-to-GPU bandwidth. Programming DL applications on AI accelerators is usually facilitated by the availability of highly optimised libraries such as TensorFlow, PyTorch and Keras.

Deep learning architectures. The basic building blocks of DNNs are artificial neurons, which are simply scalar functions of $n+1$ input variables x_i, $0 \leq i \leq n$. These inputs are processed and fed into a non-linear function, known as the activation or transfer function. DL architectures are created by combining several artificial neurons into a common topology. Conceptually, the neurons are organised into layers, where the output of one layer is fed into (at least) one other layer. Layers below the final output layer are also known as hidden layers. We now briefly review some of the most important DNN architectures in the context of bioinformatics.

Multilayer perceptrons (MLPs) are fully connected networks with a number of layers where information only flows forward along the layer stack. In the context of bioinformatics, such networks are typically not directly used, but they are leveraged to provide derived features or summaries in tabular form, such as *k*-mer histograms of sequence data. In addition, they are also often an essential part of more complex networks.

Convolutional neural networks (CNNs) rely on convolution with a small filter kernel matrix (in so-called convolution layers), the entries of which are learned during training. Because convolution allows for the replacement of values by weighted averages over their spatial neighbourhood, it enables the encoding of information about spatial structures or patterns. Convolution is often followed by an operation known as pooling, which combines the results of several convolutions into a single output. Max pooling, for example, replaces a set of convolution results with their maximum value. This allows for a certain degree of invariance: for example, in computer vision, max-pooling convolutions with the same filter kernel over several neighbouring pixels can yield translation invariance. In the context of bioinformatics, CNNs have, for example, been applied to pile-up images of MSAs in order to detect localised patterns for identifying mutations or motifs.

Variational autoencoders (VAEs) are probabilistic deep generative models designed to identify useful data encodings in an unsupervised setting. The general idea consists of simultaneously learning encoding and decoding. In training these models, the aim is to yield as faithful a reconstruction as possible while respecting desirable properties of the encoding, such as resistance to noise or sparsity. Diffusion models are another popular type of DL-based generative model that can be used to generate virtual or simulated data sets. They are based on learning a diffusion process that generates the distribution of a given data set. Generative techniques such as VAEs and diffusion models are frequently used for unsupervised learning tasks, with examples including the imputation of missing data, protein design and protein–ligand docking.

Transformers are a class of DL architecture that relies on parallel multi-head attention mechanisms. They have recently achieved remarkable performance in the field of natural language processing owing to their ability to capture complex contextual

dependencies while still allowing for efficient parallel processing on AI accelerators. Therefore, they have now replaced previous recurrent neural networks (RNNs) and long short-term memory (LSTM) models to a large extent. The development of transformer model architectures[p60] thus leverages the parallelism of GPU hardware to train highly expressive models. Consequently, they are now also making a substantial impact on a variety of applications in bioinformatics, ranging from nucleotide sequence annotation to drug discovery.

Transformer models are often categorised into two classes: scratch-trained and pre-trained. Whereas scratch-based models directly train all parameters, pre-trained models (also called foundation models[p61]) are initially trained from unlabelled data and then fine-tuned for a specific task. Large-scale foundation models such as BERT (Bidirectional Encoder Representations from Transformers) or GPT-4 have significantly advanced the state-of-the-art in language processing. We are now seeing an increasing number of exciting applications of foundation models in bioinformatics supported by the use of AI accelerators.

Sequence-based bioinformatics

Variant calling. The computational variant calling process relies on the initial mapping of sequencing reads to a reference. Traditional algorithms such as GATK apply numerous statistical models and hand-tuned heuristics to predict the likelihood of variation at each position. Using a supervised DL model for variant calling can replace the need for handcrafted heuristics by a system that has the ability to learn characteristic patterns directly from the data. The basic idea is to represent aligned reads centred around the location of interest (a candidate variant) into images and apply CNNs. This choice is motivated by the fact that inputs can be viewed as images of read pile-ups and that the complex dependence of variants on neighbouring aligned reads could be modelled by convolutional kernels.

DeepVariant[p62] pioneered this approach for calling single-nucleotide polymorphisms (SNPs) and small insertions and deletions (indels) from short-read Illumina data using the well-known Inception-v3 network architecture that outputs genotype likelihoods for the candidate location (i.e., homozygous reference, heterozygous or homozygous alternate). The authors could show that it outperformed a variety of state-of-the-art classical methods for calling variants on an unseen Ashkenazi male sample (NA24385 for the precisionFDA Truth Challenge) when trained on another sample (Genome-in-a-Bottle, NA12878). Furthermore, running DeepVariant on a traditional CPU takes about 10 hours, whereas it takes only 20 minutes with a TPU. PEPPER-Margin-DeepVariant[p63] extends DeepVariant to germline-calling results from long-read nanopore data.

Somatic variant is generally more complex than germline variant calling because variant allele frequencies (VAFs) range continuously from 0 to 1 and the available training data are limited. ClairS[p64] is the first DL-based, long-read somatic small variant caller based on a combination of an RNN and a CNN. When compared with the leading conventional methods Strelka and MuTect2, ClairS demonstrated improved performance, but required an additional 5 hours of training time on a high-end GPU.

The idea of using CNNs and RNNs for variant calling paved the way for other DL architectures that are specifically designed for different tasks. DeepConsensus[p65] improves the accuracy of sequencing reads with a gap-aware sequence transformer model. This can in turn improve the accuracy of variant calling as well as *de novo* assembly when using DeepConsensus reads. Accelerated training times have been reported on both TPUs and GPUs.

Functional annotation. The development of DL-based models also facilitates the annotation of functional elements in DNA or RNA sequences. Initial DL-based approaches for this task based on CNNs and RNNs were not able to capture complex distant contexts in sequences. Thus, recent approaches in this area are mainly based on transformer models, which are better suited for interpreting lengthy DNA/RNA sequences. DNABERT[p66] is foundation model that extends the BERT language model to DNA sequences. It is trained in two stages: a pre-training step, where an unlabelled training set is used, and a fine-tuning step, with labelled data for the desired task. DNABERT is pre-trained on *k*-mer tokens (with *k* between 3 and 6) and fine-tuned for different tasks such as transcription binding site, proximal and core promoter regions, or canonical or non-canonical splice site prediction. Training DNABERT and the newer DNABERT-2[p67] took about 17 days on 128 A100 GPUs and 14 days on eight RTX 2080Ti GPUs, respectively. Non-coding DNA determines gene expression in different cell types. Enformer[p68] combines CNNs and transformers to enable the discovery of regulatory elements located up to 100 kilobases upstream. It substantially improves gene expression prediction accuracy and is also able to learn enhancer–promoter interactions directly from DNA sequences.

Nucleotide transformer[p69] consists of a series of genome foundation models scaling up to 2.5 billion parameters. Pre-training is performed on a large collection of genomes from 850 species. Performance is evaluated based on 18 curated downstream tasks, including prediction of epigenetic histone marks, promoter–enhancer sequences, splice sites, chromatin profiles and enhancer activity. It is demonstrated that the models have acquired the ability to recognise key regulatory genomic elements. Training the largest model took 28 days on 128 A100 GPUs. Once trained, however, these foundation models can be fine-tuned at a relatively low cost: for example, fine-tuning the largest model took less than 1 hours on eight A100 GPUs.

Metagenomics. Metagenomics deals with sequencing data obtained from environmental samples. Taxonomic classification of whole-genome shotgun sequencing reads is challenging because a model needs to learn many genome-wide patterns during training, whereas only information from a short genomic fragment is used for inference.

Liang *et al*. proposed DeepMicrobes,[p70] a DL-based approach for metagenomic read classification and abundance estimation. By using a multi-layer and multi-architecture model (featuring embedding, bidirectional LSTM, self-attention and dense layers), DeepMicrobes is able to outperform state-of-the-art *k-mer*-based mapping approaches for species and genus identification in metagenomic gut data. However, this approach exhibits several shortcomings induced by the underlying LSTM architecture. The recurrent nature of such networks tends to be more difficult to train as well as less parallelizable owing to the inherently sequential nature of LSTM units, leading to high execution times

on GPUs. MetaTransformer[p71] therefore uses a scratch-based transformer model for metagenomic analysis. The transformer-encoder-based model enables efficient parallelisation on GPUs while outperforming DeepMicrobes in terms of species and genus classification abilities.

Nanopore base-calling. Long-read sequencing based on nanopore technology relies on base-calling the noisy electric current signal and requires DL to achieve high accuracy with continuously proposed new models.[p72] Chiron[p73] first uses a CNN to extract features from raw signals, then an RNN to relate such features in a temporal manner, and a connectionist temporal classification (CTC) decoder to avoid signal segmentation. Bonito[p74] substitutes the CTC decoder with a conditional random field (CRF), whereas SAcall[p75] uses transformers. RUBICON[p76] is a new framework for specialising and optimising DL-based callers. The results show that state-of-the-art throughput optimised base-callers, such as Dorado-fast, take more than 2 hours to base-call a 300-Gbps human genome at 3× coverage on an A100 GPU, which is often a bottleneck in typical bioinformatics analysis pipelines.

Structure-based bioinformatics

Protein folding. The prediction of accurate 3D protein structures from their primary amino acid sequences has long been considered one of the holy grails of bioinformatics. After more than 50 years of intensive research in this area, the automated reliable prediction of structures at atomic resolution, particularly in the absence of known structures of sufficiently close homologues, remained a hugely challenging task. In 2020, DeepMind's AlphaFold2 system[p77] clearly outperformed its competitors at CASP14 (Critical Assessment of Protein Structure Prediction 14), the 14th iteration of a biennial community-wide structure prediction challenge.[p78] Most remarkably, prediction performance for AlphaFold2 does not rely on homology, implying that, for the first time, computational structure prediction could be performed at levels of accuracy competitive with experimental structure elucidation for all classes of modelling difficulty. Since then, DeepMind has applied AlphaFold2 to a large proportion (98.5%) of the human proteome, leading to confident predictions for ≈58% of the residues,[p79] and has made the AlphaFold2 model and the predictions available to researchers.

The tremendous success of AlphaFold2 shows the impact of GPU-accelerated DL-based approaches to structural bioinformatics. Simultaneously, it demonstrates that in structural bioinformatics, DL can often not be used in a black-box manner, where topology taken over from other application areas can be expected to simply work without modification. Instead, systems such as AlphaFold2 usually employ expert knowledge and established bioinformatics approaches to inform DNNs about features of interest and suitable representations. In this manner, AlphaFold2 relies on the observation that interactions between residues often yield recoverable traces in the evolutionary record, and that direct interaction implies close spatial proximity of the residues. Sander and coworkers have shown that the evolutionary trace can indeed be used to recover spatial restraints — after removing transitive correlations — and that these restraints are often sufficient to successfully predict protein structures.[p80] In AlphaFold2, such restraints are represented as so-called pairs. The first building block of AlphaFold2, the EvoFormer module, simultaneously updates a representation of the MSA used to predict the current structure and the residue pairs. A key innovation in EvoFormer is the ability of both representations, pairs and MSA, to inform each other, leading to improved predictions.

The EvoFormer module is followed by a structure module that predicts the orientation and translation of each residue of the folded protein. Both EvoFormer and the structure module make substantial use of attention mechanisms in their respective architectures. An iteration scheme allows each step of the pipeline to inform and improve each other. Finally, the predicted structure is relaxed against the Amber force field using gradient descent. Interestingly, there are indications that the models produced by AlphaFold2 lead to noticeably worse results in protein–ligand docking studies when compared with experimentally acquired structures, despite the remarkable accuracy of the structure in general.[p81] Since the publication of AlphaFold2, several competing architectures have been proposed. Notably, both Meta AI's ESMFold[p82] and the recent trRosettaX-Single model[p83] no longer require MSAs or homologous structures as input, but instead promise to predict accurate structures from single sequences alone. Although the accuracies of these models do not yet meet AlphaFold2's achievements, they seem to improve performance on orphan proteins with no known close homologues.

Protein design. Arguably the most prominent and impressive breakthroughs in AI in recent years are related to generative application scenarios, such as text generation by OpenAI's ChatGPT and related large language models, or image generation by Stable Diffusion. Recently, these approaches have shown great promise in structural bioinformatics as well. Translating the idea of language models to proteins leads to what is predictably called protein language models. One such model class, Meta AI's ESM-family[p84],[p85] has recently been used to propose mutations to human antibodies[p86]: learning from several thousand antibody sequences, the model generated novel sequences that it thought were evolutionarily plausible. Without making any reference to the target antigen, some of the proposed sequences greatly improved affinities against several clinically relevant viruses, such as SARS-CoV-2 or Ebola. Other systems allow for a more targeted design, where desired properties or constraints of different kinds can be formulated to inform the generation process.[p87]

Whereas these systems are based on processing textual representations of protein structures, alternative approaches use diffusion models to work directly on the 3D structures. Variational diffusion models are a form of Markovian hierarchical VAE of the following form: starting with each true sample (e.g., an image), a series of encoders is trained such that each encoder samples from a Gaussian with the output of the last step (or the original sample, at the first stage) as mean, and where the last step of the chain leads to pure Gaussian noise. Simultaneously, the model trains a series of decoders that learn to reverse each individual step in the chain. Starting with randomly sampled Gaussian noise, it is then possible to sample from the distribution of the original data by running the chain of decoders. A substantial challenge in translating this approach to 3D structures of molecules lies in replacing the simple diffusion process described

above by one that respects the intrinsic geometry. The recently introduced tool FrameDiff[p88] is based on a mathematically rigorous treatment of diffusion on the special Euclidean group of rigid translations and rotations *SE*(3) through application of stochastic calculus on the appropriate Riemannian manifolds. FrameDiff allows for sampling realistic protein structures without any reference to existing ones. An alternative diffusion-based approach is less mathematically rigorous, but combines an attention-based diffusion model with a U-Net architecture that allows the conditioning of the generation process on the desired secondary structure composition.[p89]

Molecular dynamics. Force field development is one of the major challenges in MD simulations. Several popular force fields exist, each with their own peculiarities and areas of application, but depending on the chemistry involved and the scientific questions asked, none of these may be fully appropriate. Similarly, finding good parametrisations of the molecules of interest, in particular their partial charges, for the chosen force field is often difficult. AI methods can simplify both force field development and molecular parametrisation tremendously. SchNet,[p90] for instance, was successfully used to predict molecular parameters, as well as to learn potentials of mean force. The development of fully differentiable MD implementations, such as the TorchMD-framework,[p91] allows for integrating the training of force fields or parametrisations with MD code. Hybrid DL/MD approaches have also been shown to enable highly accurate simulations at reduced computational cost.[p92] In addition to force field development and molecular parametrisation, DL has been applied to learn molecular trajectories from simulation results.[p93],[p94]

Protein docking and drug discovery. DL is nowadays extensively used in different areas of drug discovery. Models such as DeepSite,[p95] which uses a 3D ConvNet-architecture to identify binding sites in proteins, can then be used in targeted docking approaches. The most common application of DL in docking is probably the design of scoring or re-scoring functions. This approach has become so popular, in fact, that at the time of writing there are several reviews on ML-based scoring functions for virtual screening,[p96] protein–ligand docking with affinity prediction[p97] or lead optimisation.[p98] ML methods have been used in scoring functions for years, either in completely empirical or semi-empirical fashion. Several studies show that for tasks such as screening or lead optimisation, methods that at least feature an ML component regularly outcompete purely classical approaches, DL-based methods have not yet demonstrated clear superiority over other, usually much simpler, learning methods.[p96],[p98] For accurate affinity prediction, the advantage of using empirical or semi-empirical methods is less clear. Unsurprisingly, performance varies strongly with the target, and the generalisation abilities of current scoring schemes have to be further investigated.[p97]

These functions can then be coupled more or less tightly to the pose optimisation scheme to yield a docking approach. GNINA 1.0[p99] is an example of a more classical approach, where DL is used only in the computation of candidate scores: the authors propose an ensemble of CNNs for re-scoring that is integrated with an AutoDock Vina implementation. On a dual 16-core 2.3 GHz Intel Xeon 5218 system with 96 GB of RAM and a 11 GB RTX 2080 Ti GPU, the CNN ensemble only added a relatively insignificant 2 seconds to the total running time of the evaluation, but greatly improved ranking quality. Other approaches also replace classical candidate search with AI. EquiBind,[p100] for instance, uses an *SE*(3)-equivariant geometric DL model to predict the rigid part of the transformation, and allows ligand flexibility while aiming to keep local atomic structures, such as bond lengths and angle bends as well as ring systems, relatively constant. Blind docking results (i.e., without a pre-supposed binding site location) compare favourably with state-of-the-art docking methods, such as the commercial GLIDE-suite, QuickVina-W or GNINA. In those tests, EquiBind was the fastest method by far, with a reported average running time of approximately 0.16 seconds per flexible ligand docking on a 16-core CPU, and 0.04 seconds on an Nvidia GTX 1060 GPU, whereas the fastest alternative, QuickVina-W, had a running time of approximately 49 seconds. Interestingly, a recent comparative study[p101] found that DL-based methods often do outperform traditional methods in the blind docking setting, but are usually less accurate than state-of-the-art traditional methods in local docking scenarios, where knowledge about the binding pocket can be used to reduce the search space. The authors also report that DL-based approaches, such as DiffDock,[p102] are some of the best currently available methods for binding pocket prediction, and can hence be used together with traditional approaches even in a blind docking scenario. However, it is already apparent that approaches based on DL will soon be able to overtake traditional docking approaches, even in scenarios where the binding pocket is fully known.

AI has also found use in virtual screening against large libraries. The DD-protocol,[p103] for instance, has been designed to screen multiple targets against libraries with $\geq$1 billion candidates. To accelerate the screening, only a small percentage of candidates is docked against the targets. A deep network is then trained on the docking results to predict docking scores from molecular descriptors. The method allowed the screening of 1.36 billion molecules from ZINC15[p104] against the SARS-CoV-2 Mpro active site in a week on a cluster consisting of 390 Intel Xeon Cores and 40 Nvidia Tesla V100 GPUs. For comparison, the more traditional protocol described in ref.[p105] required approximately 2 weeks on 10,000 CPU cores for a very similar task. Generative approaches similar to the AlphaFold2 method for protein folding have also found application in protein docking. AlphaFold-multimer[p106] is an extension of AlphaFold2 that aims at the prediction of protein complexes, even beyond simple dimeric cases. On an Nvidia A100 GPU, it typically takes a few hours per docking. A similar alternative is FoldDock,[p107] which has been applied to approximately 64,000 human protein–protein interactions, running for 5 days on an Nvidia A100 GPU system.

Third wave: quantum computing

Enabling technologies

Whereas conventional digital computers utilise bits (binary digits) as the most basic unit of information, quantum computers are based on qubits. qubits can exist in multiple states simultaneously: that is, a superposition of the base states $|0\rangle$ and $|1\rangle$.

This phenomenon allows for manipulating information at the quantum level through quantum logic gates. Multiple qubits can be correlated through quantum entanglement, which provides a superposition of any of the following 2^n states for an n-qubit system. This is a resource that is unique to QC and can lead to unprecedented speed for complex computation. By contrast, a classical computer would require an exponential number of bits to store the same data.

The design of quantum hardware is a rapidly evolving field. Typical architectures are based on a hybrid combination of a classical computer and a QPU as an additional co-processor. Two classes of QPUs are often considered, based on quantum gates and quantum annealing. IBM released Osprey in 2022, a single chip with a 433-qubit processor, and D-Wave Two is a commercially available system based on quantum annealing that features 512 qubits.

The scale of QPUs is evolving at a rapid speed, as shown on IBM's roadmap.[p9] Whereas Osprey features 433 qubits, the succeeding Condor processor will contain 1,121 qubits. A system with three Kookaburra processors totalling 4,158 qubits and quantum communication technologies is planned to be released in 2025. This shows the potential of scaling providing a clear path to 100,000 qubits and beyond in the near future.

Open Quantum Assembly Language (OpenQASM)[p108] is an example of a programming language designed for describing quantum circuits and algorithms for execution on QPUs. Because quantum hardware is still evolving, scientists also often rely on simulating quantum circuits on classical computers using optimised libraries such as cuQuantum for GPUs.

QC could in principle solve certain problems more quickly than conventional computers. Although the technology is progressing at a rapid pace, existing QC hardware systems are currently not able to provide the necessary performance for large-scale computation. Here, we therefore focus on early successes of quantum algorithms for accelerating important tasks in sequence-based and structure-based bioinformatics, which might in turn improve computational calculations in drug discovery in the future.

Quantum computing in sequence-based bioinformatics

De novo assembly of genomic DNA sequences from short sequencing reads is a highly challenging problem. Typical approaches are based on graph-theoretic methods [such as Overlap-Layout-Consensus (OLC) or de Bruijn graph methods], which in turn lead to optimal graph reductions problems that belong to the NP (nondeterministic polynomial time)-hard class of problems. Consequently, existing *de novo* assemblers rely on heuristics. Boev *et al.*[p109] and Sarkar *et al.*[p110] have recently addressed this problem based on using a quantum and quantum-inspired annealing platform. The main methodological step is to map the genome assembly problem in the framework of OLC graphs to a quadratic unconstrained binary optimisation (QUBO) problem for finding a Hamiltonian path. This can be efficiently embedded in the quantum annealing architecture or can be solved with the use of quantum-inspired optimisation algorithms. Boev *et al.* demonstrated assembly of the 4X174 bacteriophage on a D-Wave QPU.

Quantum indexed bidirectional associative memory (QiBAM)[p111] is a method for aligning short NGS reads to reference genomes for quantum accelerators. It extends Grover's quantum search algorithm to allow for errors in the alignment between reads and the reference sequence stored in quantum memory. It thus allows for approximate matches needed for read errors in genomics and a distributed search for multiple solutions over the quantum encoding of DNA sequences. This approach gives a quadratic speed increase over the classical algorithm. A full implementation of the algorithm is provided and verified using the OpenQL compiler and QX simulator framework.

MSA can be seen as a generalisation of pairwise alignment and can be solved with a generalisation of the DP algorithm using a multi-dimensional DP matrix. It has actually been shown that MSA with the sum-of-pairs scoring scheme is NP-complete. Thus, heuristic methods are used in practice. Madsen *et al.*[p112] present a binary encoding of MSA and devise a corresponding soft-constrained cost-function that enables a Hamiltonian formulation and implementation of the MSA problem with the variational quantum approximate optimisation algorithm (QAOA). The developed quantum circuit was simulated using Qiskit's software framework and run on Rigetti's Aspen-M-3 79-qubit superconducting quantum computer for small problem sizes.

Predicting the optimal RNA secondary structure (RNA folding) is another important NP-complete problem in bioinformatics. Fox *et al.*[p113] proposed a QC-based approach based on a binary quadratic model (BQM) that is derived to drive the system towards maximising the number of consecutive base-pairs while jointly maximising the average length of the stems. The QC approach is compared to a classical replica exchange Monte Carlo (REMC) algorithm programmed with the same objective function, with the Quantum Apporoximation (QA) being shown to be highly competitive at rapidly identifying low-energy solutions on the D-Wave system.

Quantum computing in structure-based bioinformatics

As described above, most methods for studying molecular structures use classical or semi-classical approximations to keep the computational effort under control. In many situations relevant for drug discovery, these approximations break down. To name just one potential challenge, most force-field-based methods are inapplicable to drugs that bind covalently to their target. In such cases, and in many others that require a detailed look at the interactions involved, quantum chemistry defines the theoretical framework to compute the quantities of interest. Classical computers, however, are famously ill-suited to computing accurate solutions for quantum chemistry problems.

A quantum computer can, in principle, solve the full configuration interaction (FCI), including even strong correlations between electrons, by diagonalising the FCI matrix. On a classical computer, however, solving the FCI scales exponentially with the number of electrons in the system, quickly rendering FCI infeasible even for small molecules. Today, a number of strategies are available to approximate the electronic structure even for large systems, which are commonly applied when the classical approximation of force-field-based molecular mechanics is unsuitable. In cases of pharmacological interest, this often includes density functional theory (DFT) computations.[p114] Because many of the systems relevant to drug design are rather weakly correlated, the DFT approximation often yields accept-

able results. However, many enzymatic reactions feature strong correlations (e.g., through the presence of transition metals in the active site), and are thus rather ill-suited to treatment with DFT. DFT is also known to underestimate long-ranged dispersion forces, which often have a significant role in drug design. Hence, there is a strong interest in going beyond such approximations for systems of pharmacologically relevant size and complexity.

Computing, say, the ground state energy for an entire system of pharmacological relevance, such as a protein–ligand complex in an aqueous environment, is still out of reach owing to the number of qubits required. However, recent developments in QC hardware have made the practical use of hybrid QM/MM (quantum mechanics/molecular mechanics) methods for practically relevant system sizes on a quantum computer a realistic near- to mid-term outlook. In such methods, different regions of the system are modelled at different levels of approximation, where the ligand and a suitably large part of the protein surrounding it are treated using quantum mechanics, while the remainder is modelled using classical molecular mechanics methods. Both parts can then be combined using one of several different strategies.[p115]

One of the biggest challenges along the path towards QC-based quantum chemistry is error correction. Although it is possible, in principle, to suppress errors or noise occurring during the computation on a quantum computer using quantum error correction, this typically entails a large increase in the number of qubits required (e.g., by combining several physical qubits into one logical qubit). Because qubits are currently a scarce resource, there is substantial interest in methods that are resistant to noise. This is often achieved by decomposing the problem into many small but rather simple quantum computations, with circuits that are shallow enough so that noise is less of an issue. Partial computations are combined by some means of classical computation. The most popular example of such an approach in quantum chemistry is the so-called variational quantum eigensolver (VQE).[p116],[p117] Here, the classical computer computes the Hamiltonian of the problem and an input state, which is described by a set of parameters θ. A series of shallow quantum circuits parameterised by θ is used to prepare a state by acting on the input state and to compute its energy, which is forwarded to a classical computer. Classical optimisation algorithms can then be used to minimise this energy. According to the variational principle, this Rayleigh–Ritz-like procedure approximates the ground state of the system. However, although this method is applicable in the presence of noise, and hence feasible on QC devices expected to be available in the near future, it is not the optimal algorithmic choice.

Blunt *et al.*[p10] performed a detailed analysis of different QC approaches to find the ground state energy of the drug ibrutinib in a complex with a tyrosine kinase. Because ibrutinib binds covalently to its target, classical molecular mechanics methods face significant challenges, and quantum approaches become highly attractive. Through a detailed analysis of different quantum algorithmic choices and error correction methods, Blunt *et al.* showed that on a fault-tolerant quantum computer, quantum phase estimation (QPE)-based methods have much lower computational complexity than VQE-approaches, at least for systems of pharmacologically relevant sizes. Using sparse qubitisation, they estimated that a quantum computer can perform the computation in a matter of days even for relatively large active spaces.

Conclusions and outlook

We have shown that modern bioinformatics tasks can benefit greatly from the use of powerful parallel accelerator technologies. Most notably, there is currently a profound transformation in the life sciences based on using AI models and GPUs. Not too long ago, accelerators were a niche HPC technology, important only for a few specialised applications. Nowadays, however, those technologies and applications are almost universally used.

We thus expect to see even more progress in this area in the near future. For example, researchers at DeepMind and Isomorphic Labs[p118] have reported significantly improved abilities of AlphaFold for protein–protein, protein–ligand and protein–nucleic acid docking, outperforming state-of-the-art approaches. At the time of writing of this review, however, this version has not yet been released. In addition, foundation models that capture the current state of understanding in specific domains such as synthetic and systems biology might also be constructed, which in turn will place new demands on accelerator technologies, experimental systems and available (well-labelled) data repositories.

As already pointed out by Richard Feynman,[p119] a quantum computer can, in principle, solve certain problems better than a classical computer. However, corresponding quantum accelerators are still at an early stage, although they are rapidly evolving. We have provided some examples of the effectiveness of QC on small tests generated with artificial data on a number of bioinformatics problems. The current limitations of QPUs still prevent their application to problem sizes in real applications. Although several theoretical results have been demonstrated in QC applied to bioinformatics, it is by no means clear whether this technology will have an essential role in the future. Nevertheless, these initial successes might pave the way to future interesting developments.

We also expect to see further research investigating the space of alternative algorithmic choices. For example, using Trotterisation instead of sparse qubitisation can lead to runtimes that can easily surpass a thousand years.[p10] Using current error-correcting codes, QPE-based algorithms would require millions of qubits for quantum chemistry. Obviously, there is a great need for further analysis of this kind, and for quantum algorithmic optimisation to drive runtimes and resource requirements down further, but it is already apparent that quantum computers have the potential to revolutionise certain areas, especially quantum chemistry in drug discovery.

CRediT authorship contribution statement

Bertil Schmidt: Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Andreas Hildebrandt:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) project number 439669440 TRR319 RMaP TP C01.

References

1. Stephens ZD et al. Big data: astronomical or genomical? *PLoS Biol*. 2015;13:e1002195.
2. Schmidt B, Hildebrandt A. Next-generation sequencing: big data meets high performance computing. *Drug Discov Today*. 2017;22:712–717.
3. Schatz MC. Biological data sciences in genome research. *Genome Res*. 2015;25:1417–1422.
4. Scheffler K et al. Somatic small-variant calling methods in Illumina DRAGEN™ Secondary Analysis. *bioRxiv*. 2023. https://doi.org/10.1101/2023.03.23.534011.
5. Owens JD et al. GPU computing. *Proc IEEE*. 2008;96:879–899.
6. Chetlur S et al. cuDNN: Efficient primitives for deep learning. *ArXiv*. 2014. https://doi.org/10.48550/arXiv.1410.0759.
7. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev*. 1958;65:386.
8. Pandey M et al. The transformational role of GPU computing and deep learning in drug discovery. *Nat Mach Intell*. 2022;4:211–221.
9. Pal S et al. Quantum computing in the next-generation computational biology landscape: from protein folding to molecular dynamics. *Mol Biotechnol*. 2024;66:163–178.
10. Blunt NS et al. Perspective on the current state-of-the-art of quantum computing for drug discovery applications. *J Chem Theory Comput*. 2022;18:7001–7023.
11. Büren F et al*Suffix Array Construction on Multi-GPU Systems*. ACM; 2019:183–194.
12. Hijma P et al. Optimization techniques for GPU programming. *ACM Comput Surv*. 2023;55:1–81.
13. Liu W et al. CUDA-BLASTP: accelerating BLASTP on CUDA-enabled graphics hardware. *IEEE/ACM Trans Comput Biol Bioinform*. 2011;8:1678–1684.
14. Ye W et al. H-BLAST: a fast protein sequence alignment toolkit on heterogeneous computers with GPUs. *Bioinformatics*. 2017;33:1130–1138.
15. Zhao K, Chu X. G-BLASTN: accelerating nucleotide alignment by graphics processors. *Bioinformatics*. 2014;30:1384–1391.
16. Liu Y et al. CUSHAW: a CUDA compatible short read aligner to large genomes based on the Burrows-Wheeler transform. *Bioinformatics*. 2012;28:1830–1837.
17. Luo R et al. SOAP3-dp: fast, accurate and sensitive GPU-based short read aligner. *PloS One*. 2013;8:e65632.
18. Wilton R, Szalay AS. Arioc: high-concurrency short-read alignment on multiple GPUs. *PLoS Comput Biol*. 2020;16:e1008383.
19. Kobus R et al*MetaCache-GPU: Ultra-Fast Metagenomic Classification*. ACM; 2021:1–11.
20. Pham M et al*Accelerating BWA-MEM Read Mapping on GPUs*. ACM; 2023:155–166.
21. Sadasivan H et al. Accelerating Minimap2 for accurate long read alignment on GPUs. *J Biotechnol Biomed*. 2023;6:13.
22. Luo R et al. BALSA: integrated secondary analysis for whole-genome and whole-exome sequencing, accelerated by GPU. *PeerJ*. 2014;2:e421.
23. O'Connell KA et al. Accelerating genomic workflows using NVIDIA Parabricks. *BMC Bioinformatics*. 2023;24:221.
24. Goswami S et al*Gpu-Accelerated Large-Scale Genome Assembly*. IEEE; 2018:814–824.
25. Awan MG et al*Accelerating Large Scale De Novo Metagenome Assembly Using GPUs*. IEEE; 2021:1–11.
26. Liu Y et al. DecGPU: distributed error correction on massively parallel graphics processing units using CUDA and MPI. *BMC Bioinformatics*. 2011;12:85.
27. Kallenborn F et al. CARE: context-aware sequencing read error correction. *Bioinformatics*. 2021;37:889–895.
28. Kallenborn F et al. CARE 2.0: reducing false-positive sequencing error corrections using machine learning. *BMC Bioinformatics*. 2022;23:227.
29. Manavski SA, Valle G. CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. *BMC Bioinformatics*. 2008;9:S10.
30. Liu Y et al. CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions. *BMC Bioinformatics*. 2013;14:117.
31. Oliveira Sandes EF et al. CUDAlign 4.0: Incremental speculative traceback for exact chromosome-wide alignment in GPU clusters. *IEEE Trans Parallel Distrib Syst*. 2016;27:2838–2850.
32. Ahmed N et al. GASAL2: a GPU accelerated sequence alignment library for high-throughput NGS data. *BMC Bioinformatics*. 2019;20:520.
33. Awan MG et al. ADEPT: a domain independent sequence alignment strategy for gpu architectures. *BMC Bioinformatics*. 2020;21:406.
34. Müller A et al. AnySeq/GPU: a novel approach for faster sequence alignment on GPUs*Proceedings of the 36th ACM International Conference on Supercomputing*. ACM; 2022:1–11.
35. Schmidt B et al. CUDASW++ 4.0: ultra-fast GPU-based Smith-Waterman protein sequence database search. *bioRxiv*. 2023. https://doi.org/10.1101/2023.10.09.561526.
36. Aguado-Puig Q et al. WFA-GPU: gap-affine pairwise read-alignment using GPUs. *Bioinformatics*. 2023;39:btad701.
37. Liu Y et al. MSA-CUDA: multiple sequence alignment on graphics processing units with CUDA*20Th IEEE International Conference on Application-Specific Systems, Architectures and Processors*. IEEE; 2009:121–128.
38. Blazewicz J et al. G-MSA—a GPU-based, fast and accurate algorithm for multiple sequence alignment. *J Parallel Distrib Comput*. 2013;73:32–41.
39. Gudyś A, Deorowicz S. QuickProbs 2: towards rapid construction of high-quality alignments of large protein families. *Sci Rep*. 2017;7:41553.
40. Liu Y et al. MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities. *Bioinformatics*. 2010;26:1958–1964.
41. Li J et al. Multicore and GPU algorithms for Nussinov RNA folding. *BMC Bioinformatics*. 2014;15(Suppl. 8):S1.
42. Stern HA, Mathews DH. Accelerating calculations of RNA secondary structure partition functions using GPUs. *Algorithms Mol Biol*. 2013;8:29.
43. Langdon W, Lorenz R. CUDA RNAfold. *Biorxiv*. 2018. https://doi.org/10.1101/298885.
44. Tchendji VK et al. A parallel tiled and sparsified Four-Russians algorithm for Nussinov's RNA folding. *IEEE/ACM Trans Comput Biol Bioinform*. 2022;20:1795–1806.
45. Salomon-Ferrer R et al. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 2. Explicit solvent particle mesh Ewald. *J Chem Theory Comput*. 2013;9:3878–3888.
46. Páll S et al. Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS. *J Chem Phys*. 2020;153 134110.
47. Phillips JC et al. Scalable molecular dynamics on CPU and GPU architectures with NAMD. *J Chem Phys*. 2020;153 044130.
48. Dietzen M et al. Large oligomeric complex structures can be computationally assembled by efficiently combining docked interfaces. *Proteins*. 2015;83:1887–1899.
49. Katchalski-Katzir E et al. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proc Natl Acad Sci USA*. 1992;89:2195–2199.
50. Kozakov D et al. The ClusPro web server for protein-protein docking. *Nat Protoc*. 2017;12:255–278.
51. Ohue M et al. MEGADOCK 4.0: an ultra–high-performance protein–protein docking software for heterogeneous supercomputers. *Bioinformatics*. 2014;30:3281–3283.
52. Macindoe G et al. HexServer: an FFT-based protein docking server powered by graphics processors. *Nucleic Acids Res*. 2010;38:W445–W449.
53. Santos-Martins D et al. Accelerating AutoDock4 with GPUs and gradient-based local search. *J Chem Theory Comput*. 2021;17:1060–1073.
54. Alhossary A et al. Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics*. 2015;31:2214–2216.
55. Hassan NM et al. Protein-ligand blind docking using QuickVina-W with inter-process spatio-temporal integration. *Sci Rep*. 2017;7:15451.
56. Ding J et al. Vina-GPU 2.0: further accelerating AutoDock Vina and its derivatives with graphics processing units. *J Chem Inf Model*. 2023;63:1982–1998.

57. Yu Y et al. Uni-dock: Gpu-accelerated docking enables ultralarge virtual screening. *J Chem Theory Comput*. 2023;19:3336–3345.
58. Markidis S et al. Nvidia tensor core programmability, performance & precision*2018 IEEE International Parallel and Distributed Processing Symposium Workshops*. IEEE; 2018:522–531.
59. Ben-Nun T, Hoefler T. Demystifying parallel and distributed deep learning: an in-depth concurrency analysis. *ACM Comput Surv*. 2019;52:65.
60. Vaswani A et al. Attention is all you need. In: von Luxburg U, ed. *Advances in Neural Information Processing Systems 30*. NIPS; 2017:5999–6009.
61. Bommasani R et al. On the opportunities and risks of foundation models. *arXiv*. 2021. https://doi.org/10.48550/arXiv.2108.07258.
62. Poplin R et al. A universal SNP and small-indel variant caller using deep neural networks. *Nat Biotechnol*. 2018;36:983–987.
63. Shafin K et al. Haplotype-aware variant calling with PEPPER-Margin-DeepVariant enables high accuracy in nanopore long-reads. *Nat Methods*. 2021;18:1322–1332.
64. Zheng Z et al. ClairS: a deep-learning method for long-read somatic small variant calling. *bioRxiv*. 2023. https://doi.org/10.1101/2023.08.17.553778.
65. Baid G et al. DeepConsensus improves the accuracy of sequences with a gap-aware sequence transformer. *Nat Biotechnol*. 2023;41:232–238.
66. Ji Y et al. DNABERT: pre-trained bidirectional encoder representations from transformers model for DNA-language in genome. *Bioinformatics*. 2021;37:2112–2120.
67. Zhou Z et al. DNABERT-2: efficient foundation model and benchmark for multi-species genome. *arXiv*. 2023. https://doi.org/10.48550/arXiv.2306.15006.
68. Avsec Ž et al. Effective gene expression prediction from sequence by integrating long-range interactions. *Nat Methods*. 2021;18:1196–1203.
69. Dalla-Torre H et al. The nucleotide transformer: building and evaluating robust foundation models for human genomics. *bioRxiv*. 2023. https://doi.org/10.1101/2023.01.11.523679.
70. Liang Q et al. DeepMicrobes: taxonomic classification for metagenomics with deep learning. *NAR Genom Bioinform*. 2020;2:lqaa009.
71. Wichmann A et al. MetaTransformer: deep metagenomic sequencing read classification using self-attention models. *NAR Genom Bioinform*. 2023;5:lqad082.
72. Pagès-Gallego M, de Ridder J. Comprehensive benchmark and architectural analysis of deep learning models for nanopore sequencing basecalling. *Genome Biol*. 2023;24:71.
73. Teng H et al. Chiron: translating nanopore raw signal directly into nucleotide sequence using deep learning. *Gigascience*. 2018;7:giy037.
74. Xu Z et al. Fast-bonito: a faster deep learning based basecaller for nanopore sequencing. *Artif Intell Life Sci*. 2021;1 100011.
75. Huang N et al. SACall: a neural network basecaller for Oxford Nanopore sequencing data based on self-attention mechanism. *IEEE/ACM Trans Comput Biol Bioinform*. 2020;19:614–623.
76. Singh G et al. RUBICON: a framework for designing efficient deep learning-based genomic basecallers. *Genome Biol*. 2024;25:49.
77. Jumper J et al. Highly accurate protein structure prediction with AlphaFold. *Nature*. 2021;596:583–589.
78. Pereira J et al. High-accuracy protein structure prediction in CASP14. *Proteins*. 2021;89:1687–1699.
79. Tunyasuvunakool K et al. Highly accurate protein structure prediction for the human proteome. *Nature*. 2021;596:590–596.
80. Marks DS et al. Protein structure prediction from sequence variation. *Nat Biotechnol*. 2012;30:1072–1080.
81. Karelina M et al. How accurately can one predict drug binding modes using AlphaFold models? *eLife*. 2023;12:RP89386.
82. Lin Z et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*. 2023;379:1123–1130.
83. Wang W et al. Single-sequence protein structure prediction using supervised transformer protein language models. *Nat Comput Sci*. 2022;2:804–814.
84. Meier J et al. Language models enable zero-shot prediction of the effects of mutations on protein function et al.. In: Ranzato M, ed. *Advances in Neural Information Processing Systems*. NIPS; 2021:29287–29303.
85. Verkuil R et al. Language models generalize beyond natural proteins. *bioRxiv*. 2022. https://doi.org/10.1101/2022.12.21.521521.
86. Hie BL et al. Efficient evolution of human antibodies from general protein language models. *Nat Biotechnol*. 2024;42:275–283.
87. Hie B et al. A high-level programming language for generative protein design. *bioRxiv*. 2022. https://doi.org/10.1101/2022.12.21.521526.
88. Yim J et al. SE(3) diffusion model with application to protein backbone generation. *arXiv*. 2023. https://doi.org/10.48550/arXiv.2302.02277.
89. Ni B et al. Generative design of *de novo* proteins based on secondary-structure constraints using an attention-based diffusion model. *Chem*. 2023;9:1828–1849.
90. Schütt KT et al. SchNet – a deep learning architecture for molecules and materials. *J Chem Phys*. 2018;148 241722.
91. Doerr S et al. TorchMD: a deep learning framework for molecular simulations. *J Chem Theory Comput*. 2021;17:2355–2363.
92. Galvelis R et al. NNP/MM: accelerating molecular dynamics simulations with machine learning potentials and molecular mechanics. *J Chem Inform Model*. 2023;63:5701–5708.
93. Tsai ST et al. Learning molecular dynamics with simple language model built upon long short-term memory neural network. *Nat Commun*. 2020;11:5115.
94. Winkler L et al. High-fidelity molecular dynamics trajectory reconstruction with bi-directional neural networks. *Mach Learn Sci Technol*. 2022;3 025011.
95. Jiménez J et al. DeepSite: protein-binding site predictor using 3D-convolutional neural networks. *Bioinformatics*. 2017;33:3036–3042.
96. Li H et al. Machine-learning scoring functions for structure-based virtual screening. *Wiley Interdiscip Rev Comput Mol Sci*. 2021;11:e1478.
97. Meli R et al. Scoring functions for protein-ligand binding affinity prediction using structure-based deep learning: a review. *Front Bioinform*. 2022;2 885983.
98. Li H et al. Machine-learning scoring functions for structure-based drug lead optimization. *Wiley Interdiscip Rev Comput Mol Sci*. 2020;10:e1465.
99. McNutt AT et al. GNINA 1.0: molecular docking with deep learning. *J Cheminform*. 2021;13:43.
100. Stärk H et al*EquiBind: Geometric Deep Learning for Drug Binding Structure Prediction*. PMLR; 2022:20503–20521.
101. Yu Y et al. Do deep learning models really outperform traditional approaches in molecular docking? *arXiv*. 2023. https://doi.org/10.48550/arXiv.2302.07134.
102. Corso G et al. DiffDock: diffusion steps, twists, and turns for molecular docking. *arXiv*. 2023. https://doi.org/10.48550/arXiv.2210.01776.
103. Gentile F et al. Artificial intelligence-enabled virtual screening of ultra-large chemical libraries with deep docking. *Nat Protoc*. 2022;17:672–697.
104. Sterling T, Irwin JJ. ZINC 15 – ligand discovery for everyone. *J Chem Inform Model*. 2015;55:2324–2337.
105. Gorgulla C et al. An open-source drug discovery platform enables ultra-large virtual screens. *Nature*. 2020;580:663–668.
106. Evans R et al. Protein Complex Prediction with AlphaFold-Multimer. *Biorxiv*. 2021. https://doi.org/10.1101/2021.10.04.463034.
107. Bryant P et al. Improved prediction of protein-protein interactions using AlphaFold2. *Nat Commun*. 2022;13:1265.
108. Cross A et al. OpenQASM 3: a broader and deeper quantum assembly language. *ACM Trans Quantum Comput*. 2022;3:12.
109. Boev A et al. Genome assembly using quantum and quantum-inspired annealing. *Sci Rep*. 2021;11:13183.
110. Sarkar A et al. QuASeR: quantum accelerated *de novo* DNA sequence reconstruction. *PloS One*. 2021;16:e0249850.
111. Sarkar A et al. QiBAM: approximate sub-string index search on quantum accelerators applied to DNA read alignment. *Electronics*. 2021;10:2433.
112. Madsen SY et al. Multi-sequence alignment using the Quantum Approximate Optimization Algorithm. *arXiv*. 2023. https://doi.org/10.48550/arXiv.2308.12103.
113. Fox DM et al. RNA folding using quantum computers. *PLoS Comput Biol*. 2022;18:e1010032.
114. Hohenberg P, Kohn W. Inhomogeneous electron gas. *Phys Rev*. 1964;136:B864–B871.
115. Cao L, Ryde U. On the difference between additive and subtractive QM/MM calculations. *Front Chem*. 2018;6:89.
116. McClean JR et al. The theory of variational hybrid quantum-classical algorithms. *New J Phys*. 2016;18 023023.
117. Peruzzo A et al. A variational eigenvalue solver on a photonic quantum processor. *Nat Commun*. 2014;5:4213.
118. Google DeepMind AlphaFold team & Isomorphic Labs team (2023) A glimpse of the next generation of AlphaFold. *Google DeepMind* https://deepmind.google/discover/blog/a-glimpse-of-the-next-generation-of-alphafold/ (published 31 October 2023; accessed 5 April 2024).
119. Feynman RP. Simulating physics with computers. *Int J Theor Phys*. 1982;21:467–488.