Application of high performance compute technology in bioinformatics

Sven Warris

Application of high performance compute technology in bioinformatics

Sven Warris

Thesis committee

Promotor
Prof. Dr D. de Ridder
Professor of Bioinformatics
Wageningen University & Research

Co-promotor
Dr J.P. Nap
Professor of Life Sciences & Renewable Energy
Hanze University of Applied Sciences Groningen

Other members
Prof. Dr B. Tekinerdogan, Wageningen University & Research
Prof. Dr R.C.H.J. van Ham, Delft University of Technology & KeyGene N.V., Wageningen
Dr P. Prins, University of Tennessee, USA
Prof. Dr R.V. van Nieuwpoort, Netherlands eScience Center, Amsterdam

Application of high performance compute technology in bioinformatics

Sven Warris

Thesis

submitted in fulfilment of the requirements for the degree of doctor

at Wageningen University

by the authority of the Rector Magnificus

Prof. Dr A.P.J. Mol

in the presence of the

Thesis Committee appointed by the Academic Board

to be defended in public

on Tuesday 22 October 2019

at 4:00 p.m. in the Aula

Sven Warris
Application of high performance compute technology in bioinformatics, 159 pages.
PhD thesis, Wageningen University, Wageningen, the Netherlands (2019)
With references, with summaries in English and Dutch
ISBN: 978-94-6395-112-8
DOI: https://doi.org/10.18174/499180

Table of contents

1 Introduction

In recent years, technological developments in the life sciences have progressed enormously. In various –omics fields, such as proteomics, metabolomics, transcriptomics and genomics, the amount of data created and the complexity of the models inferred from such data are exploding. In genomics, for example, the continuously rapid development of DNA sequencing technology is enabling researchers to (re)construct genomes at an unprecedented pace. These –omics data, commonly referred to as *big data* [1], will help advance the understanding and possible application of biological systems, provided they can be stored and analyzed properly. The growing amount of biological data and the wider scope of research questions have resulted in a large increase of bioinformatics activities. In the research field of bioinformatics biologists, computer scientists, physicists and mathematicians collaborate to integrate life sciences, information technology, data mining and modeling approaches to store, process and analyze biological data.

Major developments relevant to this thesis in both the life sciences and computer science will be outlined in the following sections focusing on genomics-related topics.

1.1 Advances in DNA sequencing technology

The history of nucleic acid sequencing is documented well [2]. It started in 1972 with the sequence of a single RNA of bacteriophage MS2, followed in 1976 by the whole RNA genome of this organism. After a number of technological advances, the DNA genome of the Epstein-Barr virus was sequenced in 1984 and the genome of the first free-living organism *Haemophilus influenzae* was published in 1995. In the late 1990's, researchers sequenced many model organisms such as yeast and *Bacillus subtilis*. These projects accelerated the use of whole-shotgun sequencing strategies, with the first draft sequence of the human genome in 2001 as notable achievement [2].

Subsequent advances in technology aimed at lower costs per base and speed-up by miniaturization and parallelization of the sequencing process. The first major commercial system was released in 2004. The 454 Life Science (later Roche) sequencing platform produced large quantities (over 20 Mb) of DNA reads of about 250 bases. The costs of sequencing a human genome with this platform dropped to about US$ 1 million [3]. In 2006, Solexa (later acquired by Illumina) introduced its HiSeq platform, capable of producing millions of reads of 35 bases in a single run. In consecutive updates the HiSeq improved significantly in terms of throughput and read length, reducing the costs per base further. The HiSeq has evolved into benchtop systems on the one hand and production-scale sequencers on the other. The benchtop MiSeq has less throughput than its big counterpart, but makes this type of sequencing technology available to many institutes. The price per base has dropped to US$ 0.0005 cents/base for the MiSeq [4]. The latter produces 1.5 Gb of data per run compared to 600 Gb/run of the HiSeq 2000. The latest update of the HiSeq, the NovaSeq [5], produces even more reads, up to 6 Tb of data within two days, providing 30 times coverage of the human genome.

Although these platforms are capable of producing large amounts of sequence data, the sequence reads are still relatively short, up to 300 bases in length, but of high quality (average 0.28% error rate [6]), limiting the applicability of the systems [7]. To be able to generate longer reads, other platforms have been introduced more recently, such as the RS II/ Sequel (Pacific BioSciences) and minION/promethION (Oxford Nanopore) systems [7]. These long-read sequencing platforms are a trade-off between the higher average and maximum length of the resulting read sets and the quality of the base-calling (average ~13% error rate) [6,7]. Two sequencing approaches that combine nucleotide data and long range contiguity information are Illumina-based: chromosome conformation capture using the Hi-C protocol [8] and synthetic long read library preparation using the 10x Genomics Chromium platform [9]. Although the underlying technologies and throughputs differ for the currently available platforms discussed above, all such modern DNA sequencing is here referred to as *next-generation sequencing* or *high-throughput sequencing*.

1.2 High-throughput sequencing technology

Various applications of high-throughput sequencing technology have been developed over the years or are being developed. Applications relevant to the work described in this thesis are outlined below.

1.2.1 *De novo* genome sequencing and assembly

Methods to determine the genome sequence of an organism when no suitable genome sequence is available for reference are referred to as *de novo* sequencing [10].

The NCBI database of complete genomes [11] currently contains 192,615 entries, of which 6,037 correspond to eukaryotic species (June 2018). Although this is a considerable amount of data, there are an estimated 8.7 million eukaryotic species, 86% of which has not yet been described or sequenced [12]. Therefore, many complete genomes will likely be sequenced *de novo* in the (near) future. An essential step in such genome sequencing efforts is the assembly of the sequence data into a contiguous sequence, *contig* for short, in a procedure known as *de novo* assembly [10]. DNA sequencers deliver data sets containing millions of (short) reads, which need to be connected into contigs in a process called assembly. The problem resembles the construction of a giant jigsaw puzzle of which the image is unknown. The main challenge for *de novo* assemblers, such as Canu [13] and many others [10], are the repetitive parts of a genome, which can be several kilobases long [14]. The main issue with short read lengths is that they fail to span repeat regions longer than the read length. This limits their application in *de novo* assembly [10,15]. To unambiguously assemble reads into contigs, repeats need to be spanned by reads; short reads hence do not suffice [10,15–17].

The Pacific Biosciences Sequel and Oxford Nanopore sequencers can create long reads, but these contain too many errors (~13%) [6,7] to be handled effectively and efficiently by currently available assembly software for large genomes. For small genomes (1-40 Mb), such as bacteria and most fungi, long read technologies have been shown to allow near-complete to fully complete *de novo* assemblies [18]. Hybrid approaches, where data from different sequencing platforms are combined, are gradually becoming commonplace, especially for large genomes, and will replace single-technology based approaches as the standard for *de novo* genome assembly [19,20]. In hybrid assembly short reads are used to create high-quality contigs and long reads are used to fill the gaps between these contigs and bridge the repetitive parts [17,21]. A useful addition to the challenges of hybrid *de novo* assembly is the BioNano Genomics Irys system and its successor the Saphyr [22]. In this technology of (high-throughput) optical mapping, very long molecules (high molecular weight DNA) are fluorescently labeled, separated and ordered [23]. The BioNano platforms are used for scaffolding genome assemblies as these molecules offer long-range contiguity information [24]. When the resulting assembly still contains scaffolds (sets of linked, oriented contigs) rather than full chromosomes, linked-read technology and/or optical mapping can be used to arrange the scaffolds, perform gap size predictions and visualize other structural information [24].

De novo assembly of large (>100 Mb) genomes is computationally complex. Several approaches are usually combined, including De Bruijn graphs for the initial assembly [25], read mapping for quality control [26] and filling of gaps between contigs [27]. Such approaches put strains on computational resources such as memory and CPU power [28]. Moreover, the size of the sequence data sets continues to grow, further challenging the computational requirements of assembly [29]. As a result, IT developments have difficulties keeping up with the speed of growing computational demands. New ways of tackling the assembly process are constantly being developed to allow proper *de novo* assembly, for example by removing the compute intensive error correction phase [30,31] or by changing the order in which the assembly process takes place [32].

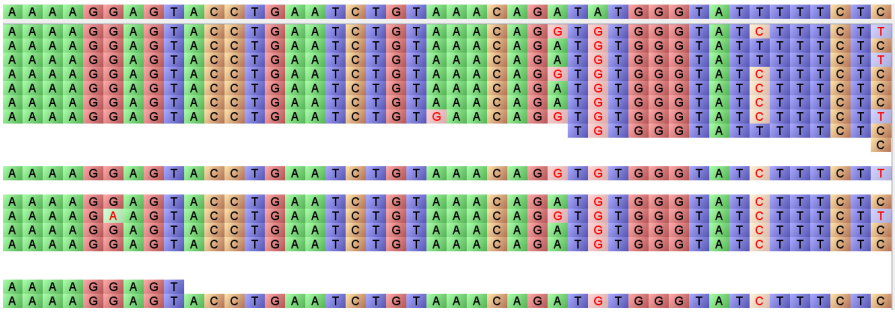

Figure 1.1. Read mapping in the process of genome resequencing. Each sequence read is mapped to the available reference genome (top line) to find the location(s) of the reads, allowing for mismatches and gaps. Computationally, reads are assembled in the new genome sequence. This (partial) image was created with Tablet [33].

1.2.2 Genome resequencing

If the genome of a species is available, there is no need to perform a *de novo* assembly. The available genome sequence is used as reference (similar to the image of a puzzle) and reads are aligned to it (Figure 1.1). This process is called read mapping [34] and is an important step in genome resequencing. With this approach, less data is required than for *de novo* assembly, but repetitive parts of the reference are still problematic: a short read from a repetitive part will map to more than one location. This ambiguity issue increases if mismatches and gaps are allowed. Also genetic variation, (poly)ploidy and sequencing errors make read mapping more challenging. After reads are mapped to the reference, a consensus is extracted, giving the reassembled genome of the sequenced organism. Several software tools assist in such genome assembly [34–37], such as BWA [38,39], Bowtie2 [40] and Minimap2 [41].

1.2.3 Variant discovery

At the DNA level, each individual organism differs from other organism of the same species. Such differences determine in large part how the organism (dis-) functions. It is therefore important to identify these small differences, or variants, between individuals. For variant discovery, reads are mapped to a reference genome prior to statistical analyses [42] (Figure 1.2). Changes with respect to the reference may indicate changes in biological function. Tools such as BWA [38] or Minimap2 [41] are used to map reads, but in these tools the number of mismatches, insertions or deletions allowed in a read [34,37,43] is limited. As a result, other variants, such as larger insertions or deletions require more accurate algorithms for read mapping (Chapters 3 and 4).

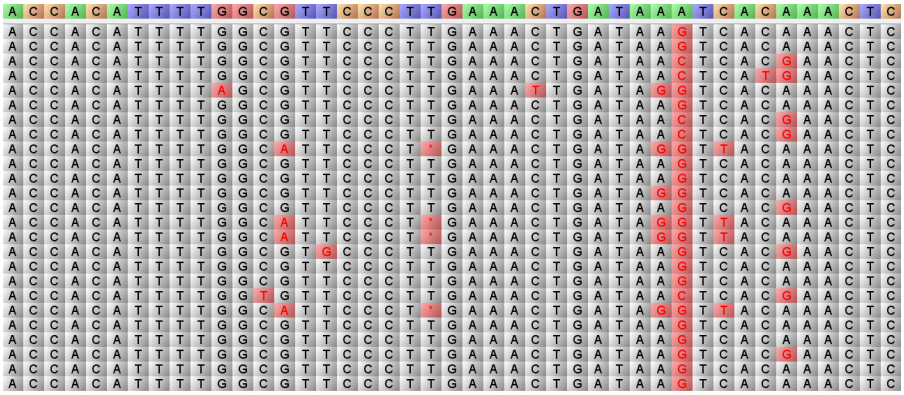

Figure 1.2. Variant detection by mapping reads to a reference genome. The reads are highly similar, as shown by the many matching bases (grey). Nucleotides colored in red indicate where the reads differ from the reference sequence, with a star (*) indicating a gap. The G/C variant is an obvious single nucleotide polymorphism (SNP) and further statistical analysis will determine whether the other variants are read errors or true variants. This image was created with Tablet [33].

1.2.4 RNA sequencing

High-throughput sequencing technology also allows determining the RNA content of a cell (i.e. the transcriptome) [44]. As RNA sequences encode either protein sequences (messenger RNA or mRNA) or are regulatory molecules, the resulting data presents a snapshot of the ongoing processes within a living cell. Such RNA sequencing indicates for example transcriptionally active genes, whereas the number of sequences found indicates transcription levels. Tools similar to those used for read mapping outlined above are used to find the location of the RNA in the genome sequence, and to identify the corresponding gene. The phenomenon of splicing [45] and occurrence of splice variants make this mapping more challenging. The PacBio and Oxford Nanopore platforms allow sequencing the full-length mRNA, which increases the likelihood to include splicing variants [46,47]. The 10X Chromium platform and other technologies allow for RNA sequencing at a single-cell level [48], further advancing the accuracy of RNA analysis.

Besides mRNA sequencing, other types of RNA are of interest, for example long non-coding RNA [49], and small interfering RNA (siRNA) [50]. In recent years it has become clear that siRNAs play important roles in regulating transcription and translation [50]. For example, small RNA molecules bind to messenger RNA and block transcription or enable degeneration of messenger RNA before transcription takes place. A particular class of small interfering RNA sequences are microRNAs (or miRNAs) [51]. The biogenesis of miRNAs includes a pre-miRNA sequence of 50-600 bases (Figure 1.3) which is shortened to an active sequence of 17-22 bases.

When sufficiently expressed, pre-miRNA molecules can be detected through short-read RNA sequencing [44] that targets transcribed DNA. Expression levels of pre-miRNAs tend however to be low and to identify all possible pre-miRNA genes in a genome, the entire genome of the organism should be analyzed. Based on the genome sequence, miRNA genes and their targets are predicted [51]. Such predictions are compute-intensive and require considerable amounts of resources.

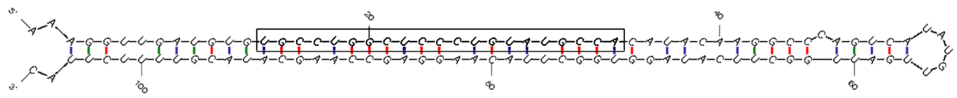

Figure 1.3. 2D structure of a pre-miRNA molecule. The pab-MIR160a [52] RNA molecule is folded in a hairpin structure, with a loop on the right. Such a hairpin structure is indicative for pre-miRNA molecules. The location of the mature miRNA is indicated by the box. Structure and image were created with the MFold website [53].

1.3 Advances in computer science

The developments in the life sciences outlined above demand significantly growing compute- and computer-intensive resources to store, process and analyze the data generated. In recent years, the field of computer science has seen several major developments relevant for these challenges. These will be outlined below.

Arguably the first computer in the world, the EDSAC1, was built in 1949 in the UK [54]. As early as 1950 the first biological application ran on this computer: the calculation of gene frequencies by the famous statistician Ronald A. Fisher [55]. One could therefore argue that the field of bioinformatics was born in 1950. From a modern perspective, the power of computers in those days was very low but it has since doubled about every 18 months, an observation known as Moore's Law [56]. Supercomputers are nowadays installed all over the world and are used for data and compute-intensive research fields, such as medical science, astronomy, climate research, defense, national security and (population) genetics. BGI in China, nowadays the largest DNA sequencing facility in the world, has over 212 TeraFLOP/s of computing power available to process its output [57]. Such supercomputers are not easily accessible for a smaller research institute. Not only the costs of purchasing and running a supercomputer prevent many organizations to establish one, supercomputers also require dedicated software and highly skilled personnel for development and maintenance. However, nowadays several new technology platforms provide (smaller) research organizations with relatively low-cost, yet high-performance, computing power, such as grids, clouds and general-purpose graphics processing units (GPGPU).

1.3.1 Grid computing

Desktop computers and servers are common-place in life science research organizations such as universities and other higher-education and/or research institutes. Such computers run a variety of operating software platforms, from Windows to Linux, and are used for many different tasks: word processing, calculations in spreadsheets, sequence alignments, protein modeling, etc. Desktop work such as word processing does not require much computing power and users typically require computation time at most 8 hours a day. Therefore, the computer is idle most of the day. To be able to use such computers for research tasks, dedicated software is available to create a so-called computer grid. For example IBM (www.ibm.com), Globus (www.globus.org) or HTCondor (research.cs.wisc.edu/htcondor/) offer such software. Installing one of these packages on the desktops and servers creates a computer grid of tens up to thousands of nodes, limited only by the nodes available. When desktops are idle, they are automatically used for desired applications. Provided issues with security and privacy are tackled satisfactorily, these grid technologies give easy and affordable access to low-cost, high-throughput computing facilities without the need to rewrite software and/or the purchase of (costly) additional hardware. In bioinformatics, grids are used and useful for computationally intensive tasks, such as protein folding [58] or BLAST searches [59]. In Chapter 2 of this thesis, an example of the use of an HTCondor grid for RNA sequence calculations is given.

1.3.2 Cloud computing and storage

In case data is dynamic or larger than a standard desktop hard drive of several TB can handle, local storage and grid computing become ineffective: changes in the data need to be sent over the network to each of the nodes or entire data sets have to be redistributed. Companies such as Google and Amazon developed cloud technology [60] to deal with huge data volumes and continuously changing and highly dynamic data. Three design principles play a role in cloud technology: (a) low-end and cheap hardware, (b) cloud storage and (c) cloud computing [61].

The use of low-end and cheap hardware is similar to the hardware used in grids (see previous section). It makes cloud technology easily accessible and affordable for relatively small users. Also, costs remain manageable even when cloud technology is deployed on a large scale. Cloud storage is the concept of storing data locally, but not everything is stored on every node [60]. Data distributed over the network is split up in large blocks and each block is stored on two or three nodes simultaneously. In case a node fails, all data are still available. Cloud storage also reduces the need to send data many times over the network. Cloud computing is the concept to calculate locally what is stored locally. Each node performs only the calculations on the data which is stored on that node. There is almost no network traffic required to fetch the data. Intermediate results are stored locally and only the end result is gathered at a central point. Google uses cloud technology

to search through millions of web pages millions of times a day, reporting relevant results to the user within a tenth of a second. The underlying computational model is called MapReduce [62]. Detailed description of MapReduce [62] is beyond the scope of this introduction, yet its performance is obviously interesting for the data volumes and computations of bioinformatics [61,63].

Bioinformatics data sets and analyses are suitable to be placed in the cloud, either in a private cloud or in one of a commercial enterprise such as Amazon.com, where researchers can rent cloud space and computer time [64]. Data is stored and accessed in the cloud in a different way than in traditional network storage systems such as Network File System (NFS) and New Technology File System (NTFS), requiring a (partial) redesign of existing applications. If an application requires access to a cloud-based database, the database model and data transfer have to be redesigned. In case of a MapReduce-based approach, the entire data processing model of the application needs to be reconsidered. To date, only a limited number of bioinformatics applications are available for use in the cloud, such as BLAST [65], BAM sequence alignment processing [66] and CloudBrush [67]. It is expected that this number will rise rapidly in the future, because frameworks such as ADAM [68] and Apache Spark [69] will take away most of the developmental efforts from bioinformatics researchers, making them more accessible.

1.3.3 General Purpose Graphics Processing Units (GPGPU)

Grid and cloud technology focus on networked use of many computer systems. However, single systems offer possibilities for high-performance computing as well. The Central Processing Unit (CPU) of a computer performs most of the logical instructions. In addition, contemporary desktop computers hold another processor: the Graphics Processing Unit (GPU). This GPU performs the calculations necessary to display information on the screen based on using a large number (hundreds) of small computing elements. Over the years GPUs have become complex processors able to generate high-resolution gaming environments for a realistic user experience. For many specific applications the GPU is faster than the CPU [70]. The development of CPU and GPU speed, expressed in floating point operations per second (FLOP/s), is depicted in Figure 1.4. In 2005 the GPU became faster than the CPU in single precision calculations and from 2008 on it has been also faster in double precision calculations. The increase in GPU speed is also higher than the speed of a CPU, with the Sandy Bridge CPU reaching ~90 GFLOP/s and the Geforce GTX 580 reaching ~1500 GFLOP/s in 2010. The most recently released GPU, the high-end NVIDIA Tesla V100 (2018), has a peak performance of 15.7 TFLOP/s. With the release of the CUDA general-purpose programming language by NVIDIA Corporation (www.nvidia.com) in 2007 and the open standard OpenCL (www.khronos.org/opencl) in 2008, the computational power of the GPU became accessible for other purposes than graphics processing. This makes the GPU a General-Purpose Graphics Procession Unit (GPGPU, www.gpgpu.org).

GPUs have two important advantages over CPUs: they deliver more FLOP/s per unit cost and they are more energy efficient: the FLOP/s/Watt is higher for a GPU than for a CPU [72,73]. In a research setting, GPUs therefore deliver low-cost, high-performance hardware, giving smaller research organizations access to computing power traditionally reserved for supercomputers, i.e. they help 'democratizing' supercomputing. GPUs are readily installed in standard desktop computers and specialized servers are available capable of holding several GPUs. Moreover, CPU vendors such as Intel recognize the increased use of GPU technology and have started integrating GPU hardware within their CPU (HD Graphics) and building accelerator cards with many CPU cores (Xeon Phi), similar to a GPU card. Combined with grid and/or cloud technology, GPGPU technology increases available computing resources even further.

A limitation in the use of a GPU is that the hardware has a completely different design than that of a CPU and performs calculations in parallel [71]. As the hardware is inherently parallel, it needs to be programmed as such [71]. Each individual processing unit on a GPU is slower than that of a CPU but there are a large number of these small units on a GPU. To harness the power of the GPU, any application needs to be re-designed, considering concurrency, thread synchronization and race conditions [74]. Resources on a GPU are also limited [71]. Currently only high-end Tesla high-performance GPUs and GeForce 20 series GPUs have 8 GB of main memory or more, which is the de facto standard for a basic desktop computer. Local memory is limited to 16 KB or 32 KB per processing unit. Making full use of the fast but limited memories on a GPU is therefore challenging, requiring trade-offs between storage efficiency (space) and the computational requirements of data conversion (time) [74].

Any implementation of a bioinformatics algorithm for a CPU has to be rewritten (in part) to make optimal use of the GPU [75]. Such parallel programming, often done in the C/C++-based CUDA or OpenCL programming language [76], is a complex task. Fortunately, more and more developers are using GPGPU solutions [75], as a result of which commonly used programming languages offer integration with CUDA and/or OpenCL. Applications written in for example Java, Python or Matlab can be extended through libraries that make use of GPGPU technology [77,78]. These recent developments have two advantages. First, applications need not be rewritten completely, but only the GPGPU part needs to be ported to CUDA/OpenCL. Second, the community of developers for, for example, Python is much larger and offers additional support for faster porting to GPUs. As a result, GPGPU computing is becoming better accessible for bioinformatics application development. Examples of GPGPU computing in bioinformatics include multiple sequence alignment, computing gene regulatory networks and many others [79–82], including the approaches towards sequence alignment in Chapters 3 and 4 of this thesis.

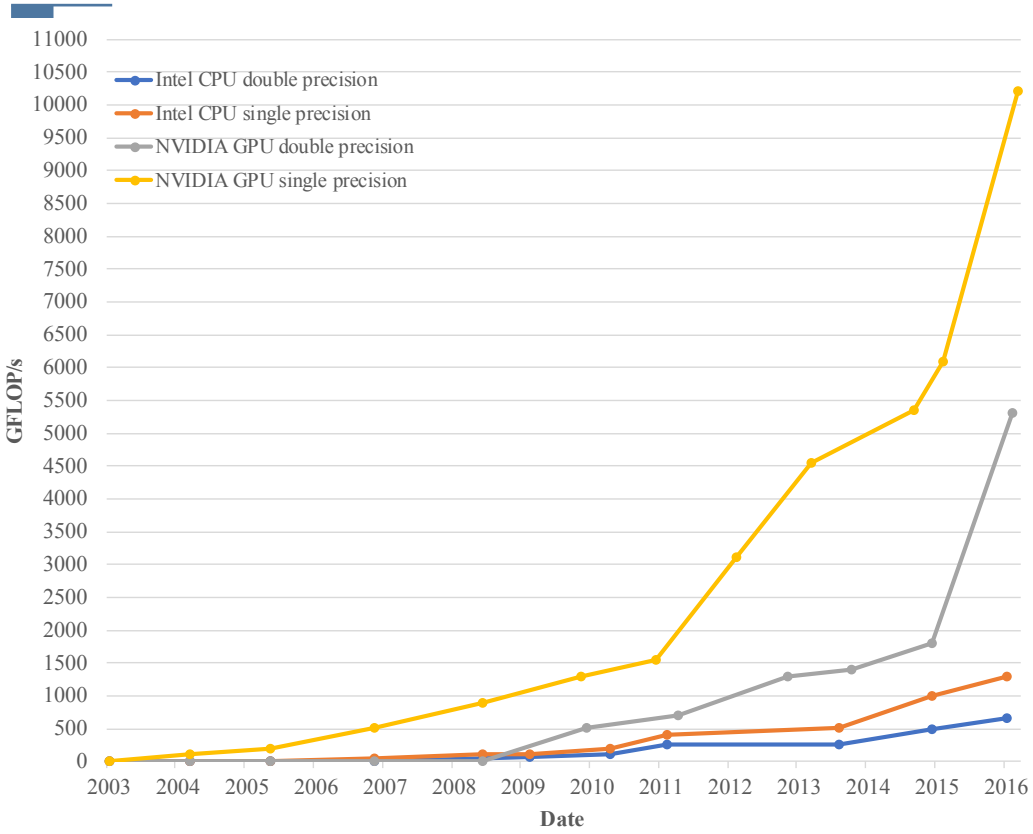

Figure 1.4. Development of capability of Intel CPUs and NVIDIA GPUs in giga floating point operations per second (GFLOP/s) over years. From [71].

1.3.4 Technologies for storing and processing data

Computational demands in other fields, such as social media, have accelerated the development of new database technologies, other than the relational (SQL) databases [83]. These new technologies are usually referred to as NoSQL databases [84]. HBase [85], for example, stores information in a key-value structure comparable to a dictionary or hash map data structure in a programming language: a tuple-store. Neo4j [86], amongst others, stores data using an object-relationship-object data structure and is called a graph or triple store. Such data stores are highly scalable [84] and therefore attractive for use in bioinformatics research [85]. As data is structured in a different way than in relational databases, algorithms and tools using such tuple/triple stores need to adjust their approach to data storage, retrieval and analysis. Many datasets in biology and bioinformatics will however benefit from storage and analyses using NoSQL paradigms to advance biological research. Examples of such developments are presented in Chapter 6 of this thesis.

1.4 Outline of this thesis

In this thesis it is demonstrated that the incorporation of new developments in information technology with respect to both hardware and software allows for larger-scale analysis in bioinformatics at reasonable compute- and computer-intensive investments without the need for supercomputer infrastructure. These developments accommodate the continuous growth of data and data analyses in the life sciences and bioinformatics. State-of-the art technologies, such as GPUs and NoSQL databases, are combined with the latest developments and issues in DNA sequencing technology for bioinformatics analyses. In most cases, proof-of-principle is given for one or a few species or data types, yet the results outline the broader impact of the approaches developed.

In Chapter 2 it is demonstrated that genome-wide prediction of miRNA candidates is feasible on a much larger scale than before owing to the development of a large pre-calculated database of 2D RNA structure predictions. A database of millions of pre-calculated minimum free energy (MFE) values was created and used to estimate the MFE of any candidate miRNA sequence within a much shorter time frame than existing approaches needed.

Chapter 3 shows the novel implementation of the widely used golden standard for sequence alignment, the Smith-Waterman (SW) algorithm, on GPUs. The Parallel Smith-Waterman Alignment Software (PaSWAS) allows parallel aligning DNA, RNA and protein sequences on a large scale on low-cost commodity hardware, while allowing inspection and selection of alignments. To the best of our knowledge, this is the first parallel implementation of the SW algorithm that allows such inspection and selection. PaSWAS is written in C and CUDA and it requires expert knowledge and skills to use, maintain and integrate PaSWAS in other software.

Chapter 4 presents the need and development of a Python-based implementation of PaSWAS to improve the overall applicability and attraction of parallel SW analyses: pyPaSWAS. It presents a more user- and developer-friendly interface in Python, builds better on standard libraries such as bioPython, pyCUDA and pyOpenCL and supports different input and output formats. In pyPaSWAS, OpenCL versions for GPUs and CPUs are integrated to further support a wider variety of hardware.

In Chapter 5, the pyPaSWAS approach is used to detect artificial chimeric sequences in long reads. These artefacts are unintentionally introduced when Whole Genome Amplification (WGA) is used in case only limiting amounts of DNA of the biological sample is available for sequencing. Such chimeric reads hamper *de novo* assembly and read mapping. The procedure for chimera detection and read-correction improves read mapping and *de novo* assembly to the point that WGA becomes feasible for accurate long-read sequencing technology in case of limiting DNA amounts.

Chapter 6 focuses on comparative functional genomics using graph databases and details the newly developed Cytoscape plug-in for querying, visualization and analyses of the type of graph structures generated in the comparative genomics. The thesis concludes with a general discussion (Chapter 7) evaluating the implications of this work in the context of future research and development issues in the life and computer sciences.

1.5 References

1. Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron MJ, et al. Big Data: astronomical or genomical? PLoS Biol. 2015;13.
2. Heather JM, Chain B. The sequence of sequencers: The history of sequencing DNA. Genomics. 2016;107:1–8.
3. Human Genome Sequencing Consortium International. Finishing the euchromatic sequence of the human genome. Nature. 2004;431:931–45.
4. Loman NJ, Misra R V, Dallman TJ, Constantinidou C, Gharbia SE, Wain J, et al. Performance comparison of benchtop high-throughput sequencing platforms. Nat. Biotechnol. 2012;30:434–9.
5. Illumina. NovaSeq 6000 Sequencing System [Internet]. Available from: https://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/novaseq-6000-system-specification-sheet-770-2016-025.pdf
6. Quail MA, Smith ME, Coupland P, Otto TD, Harris SR, Connor TR, et al. A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC Genomics. 2012;13:341.
7. van Dijk EL, Jaszczyszyn Y, Naquin D, Thermes C. The third revolution in sequencing technology. Trends Genet. 2018;34:666–81.
8. Belton J-M, McCord RP, Gibcus JH, Naumova N, Zhan Y. Hi–C: A comprehensive technique to capture the conformation of genomes. Methods. 2012;58:268–76.
9. Zheng GXY, Lau BT, Schnall-Levin M, Jarosz M, Bell JM, Hindson CM, et al. Haplotyping germline and cancer genomes with high-throughput linked-read sequencing. Nat. Biotechnol. 2016;34:303–11.
10. Khan AR, Pervez MT, Babar ME, Naveed N, Shoaib M. A comprehensive study of *de novo* genome assemblers: current challenges and future prospective. Evol. Bioinforma. 2018;14.
11. NCBI. NCBI Genome website [Internet]. Available from: http://www.ncbi.nlm.nih.gov/genome
12. Mora C, Tittensor DP, Adl S, Simpson AGB, Worm B. How many species are there on Earth and in the ocean? PLoS Biol. 2011;9:e1001127.
13. Koren S, Walenz BP, Berlin K, Miller JR, Bergman NH, Phillippy AM. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Genome Res. 2017;27:722–36.
14. Jiao W-B. The impact of third generation genomic technologies on plant genome assembly. Curr. Opin. Plant Biol. 2017;36:64–70.
15. Alkan C, Sajjadian S, Eichler EE. Limitations of next-generation genome sequence assembly. Nat. Methods. 2011;8:61–5.
16. Zhang W, Chen J, Yang Y, Tang Y, Shang J, Shen B. A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. PLoS One. 2011;6.
17. Schatz MC, Delcher AL, Salzberg SL. Assembly of large genomes using second-generation sequencing. Genome Res. 2010;20:1165–73.
18. Ee R, Lim Y-L, Yin W-F, Chan K-G. *De novo* assembly of the quorum-sensing *Pandoraea sp.* Strain RB-44 complete genome sequence using PacBio single-molecule real-time sequencing technology. Genome Announc. 2014;2.
19. Gao Y, Wang H, Liu C, Chu H, Dai D, Song S, et al. De novo genome assembly of the red silk cotton tree (*Bombax ceiba*). Gigascience. 2018;7.
20. Yin D, Ji C, Ma X, Li H, Zhang W, Li S, et al. Genome of an allotetraploid wild peanut *Arachis monticola*: a *de novo* assembly. Gigascience. 2018;7.

21. Diguistini S, Liao NY, Platt D, Robertson G, Seidel M, Chan SK, et al. *De novo* genome sequence assembly of a filamentous fungus using Sanger, 454 and Illumina sequence data. Genome Biol. 2009;10:R94.
22. Wang J, Wing Chun Pang A, Lam ET, Andrews W, Anantharaman T, Hastie A, et al. Building high quality, chromosome scale, *de novo* genome assemblies by scaffolding next generation sequencing assemblies with Bionano genome maps. AGBT 2018.
23. Tang H, Lyons E, Town CD. Optical mapping in plant comparative genomics. Gigascience. 2015;4:3.
24. Chaney L, Sharp AR, Evans CR, Udall JA, Allen P, Caicedo AL, et al. Genome mapping in plant comparative genomics. Trends Plant Sci. 2016;0:545–7.
25. Compeau PEC, Pevzner PA, Tesler G. How to apply de Bruijn graphs to genome assembly. Nat. Biotechnol. 2011;29:987–91.
26. Walker BJ, Abeel T, Shea T, Priest M, Abouelliel A, Sakthikumar S, et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. PLoS One. 2014;9.
27. English AC, Richards S, Han Y, Wang M, Vee V, Qu J, et al. Mind the gap: upgrading genomes with Pacific Biosciences RS long-read sequencing technology. PLoS One. 2012;7.
28. Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, et al. *De novo* assembly of human genomes with massively parallel short read sequencing. Genome Res. 2010;20:265–72.
29. Zerbino DR, Birney E. Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs. Genome Res. 2008;18:821–9.
30. Vaser R, Sović I, Nagarajan N, Šikić M. Fast and accurate *de novo* genome assembly from long uncorrected reads. Genome Res. 2017;27:737–46.
31. Li H. Minimap and miniasm: fast mapping and *de novo* assembly for noisy long sequences. Bioinformatics. 2016;32:2103–10.
32. Nowoshilow S, Schloissnig S, Fei J-F, Dahl A, Pang AWC, Pippel M, et al. The axolotl genome and the evolution of key tissue formation regulators. Nature. 2018;554:50–5.
33. Milne I, Bayer M, Cardle L, Shaw P, Stephen G, Wright F, et al. Tablet--next generation sequence assembly visualization. Bioinformatics. 2010;26:401–2.
34. Schbath S, Martin V, Zytnicki M, Fayolle J, Loux V, Gibrat J-F. Mapping reads on a genomic sequence: an algorithmic overview and a practical comparative analysis. J. Comput. Biol. 2012;19:796–813.
35. Li R, Yu C, Li Y, Lam T-W, Yiu S-M, Kristiansen K, et al. SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics. 2009;25:1966–7.
36. Weese D, Emde A-K, Rausch T, Döring A, Reinert K. RazerS--fast read mapping with sensitivity control. Genome Res. 2009;19:1646–54.
37. David M, Dzamba M, Lister D, Ilie L, Brudno M. SHRiMP2: sensitive yet practical SHort Read Mapping. Bioinformatics. 2011;27:1011–2.
38. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics. 2009;25:1754–60.
39. Drozd A, Maruyama N. Fast GPU read alignment with Burrows Wheeler transform based index. Perform. Eval. 2011;1–4.
40. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat. Methods. 2012;9:357–9.
41. Li H, Birol I. Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics. 2018;34:3094–100.
42. Van der Auwera GA, Carneiro MO, Hartl C, Poplin R, del Angel G, Levy-Moonshine A, et al. From FastQ data to high-confidence variant calls: the genome analysis toolkit best practices pipeline. Curr. Protoc. Bioinformatics. 2013.
43. Bao S, Jiang R, Kwan W, Wang B, Ma X, Song Y-Q. Evaluation of next-generation sequencing software in mapping and assembly. J. Hum. Genet. 2011;56:406–14.
44. Mortazavi A, Williams BA, McCue K, Schaeffer L, Wold B. Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nat. Methods. 2008;5:621–8.
45. Herzel L, Ottoz DSM, Alpert T, Neugebauer KM. Splicing and transcription touch base: co-transcriptional spliceosome assembly and function. Nat. Rev. Mol. Cell Biol. 2017;18:637–50.
46. Rhoads A, Au KF. PacBio sequencing and its applications. Genomics, proteomics & bioinformatics. 2015;13:278–89.

47. Garalde DR, Snell EA, Jachimowicz D, Sipos B, Lloyd JH, Bruce M, et al. Highly parallel direct RNA sequencing on an array of nanopores. Nat. Methods. 2018;15:201–6.

48. Haque A, Engel J, Teichmann SA, Lönnberg T. A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. Genome Med. 2017;9:75.

49. Mercer TR, Dinger ME, Mattick JS. Long non-coding RNAs: insights into functions. Nat. Rev. Genet. 2009;10:155–9.

50. Ipsaro JJ, Joshua-Tor L. From guide to target: molecular insights into eukaryotic RNA-interference machinery. Nat. Struct. Mol. Biol. 2015;22:20–8.

51. Sarker R, Bandyopadhyay S, Maulik U. An overview of computational approaches for prediction of miRNA genes and their targets. Curr. Bioinform. 2011;6:15.

52. Griffiths-Jones S, Saini HK, van Dongen S, Enright AJ. miRBase: tools for microRNA genomics. Nucleic Acids Res. 2008;36:D154–8.

53. Zuker M. Mfold web server for nucleic acid folding and hybridization prediction. Nucleic Acids Res. 2003;31:3406–15.

54. Campbell-Kelly M. Programming the EDSAC: early programming activity at the university of cambridge. IEEE Ann. Hist. Comput. 1980;2:7–36.

55. Fisher RA. Gene frequencies in a cline determined by selection and diffusion. Biometrics. 1950;6:353–61.

56. García-Risueño P, Ibáñez PE. A review of High Performance Computing foundations for scientists. Int. J. Mod. Phys. C. 2012;33.

57. BGI [Internet]. Available from: https://www.bgi.com/global/

58. Dill KA, MacCallum JL. The protein-folding problem, 50 years on. Science. 2012;338:1042–6.

59. Krishnan A. GridBLAST: a Globus-based high-throughput implementation of BLAST in a Grid computing framework. Concurr. Comput. Pract. Exp. 2005;17:1607–23.

60. Hadoop - Apache Software Foundation project home page [Internet]. Available from: http://hadoop.apache.org/

61. Thakur RS, Bandopadhyay R, Chaudhary B, Chatterjee S. Now and next-generation sequencing techniques: future of sequence analysis using cloud computing. Front. Genet. 2012;3:280.

62. Rosen J, Polyzotis N, Borkar V, Bu Y, Carey MJ, Weimer M, et al. Iterative MapReduce for large scale machine learning. CoRR. 2013;abs/1303.3.

63. Shanker A. Genome research in the cloud. OMICS. 2012;16:422–8.

64. Amazon. Amazon Cloud [Internet]. Available from: http://aws.amazon.com/ec2/

65. Matsunaga A, Tsugawa M, Fortes J. CloudBLAST: combining MapReduce and virtualization on distributed resources for bioinformatics applications. 2008 IEEE Fourth Int. Conf. eScience. IEEE; 2008. p. 222–9.

66. Niemenmaa M, Kallio A, Schumacher A, Klemelä P, Korpelainen E, Heljanko K. Hadoop-BAM: directly manipulating next generation sequencing data in the cloud. Bioinformatics. 2012;28:876–7.

67. Chang Y-J, Chen C-C, Ho J-M, Chen C-L. De novo assembly of high-throughput sequencing data with cloud computing and new Operations on string graphs. 2012 IEEE Fifth Int. Conf. Cloud Comput. IEEE; 2012. p. 155–61.

68. Genomics BD. ADAM. Available from https://github.com/bigdatagenomics/adam

69. Guo R, Zhao Y, Zou Q, Fang X, Peng S. Bioinformatics applications on Apache Spark. Gigascience. Oxford University Press; 2018;7.

70. Lee VW, Hammarlund P, Singhal R, Dubey P, Kim C, Chhugani J, et al. Debunking the 100X GPU vs. CPU myth. ACM SIGARCH Comput. Archit. News. 2010;38:451.

71. NVIDIA CUDA programming guide. NVIDIA Corporation; 2017.

72. Hamada T, Benkrid K, Nitadori K, Taiji M. A comparative study on ASIC, FPGAs, GPUs and general purpose processors in the O(N^2) gravitational N-body simulation. 2009 NASA/ESA Conf. Adapt. Hardw. Syst. Ieee; 447–52.

73. Hobiger T, Kimura M, Takefuji K, Oyama T, Koyama Y, Kondo T, et al. GPU Based Software Correlators - Perspectives for VLBI2010. IVS 2010 Gen. Meet. Proc. 2010;40–4.

74. Kirk DB, Hwu WW. Programming massively parallel processors. ISBN 978-0-12-811986-0. 2017.

75. Farber RM. Topical perspective on massive threading and parallelism. J. Mol. Graph. Model. 2011;30:82–9.

76. Demidov D, Ahnert K, Rupp K, Gottschling P. Programming CUDA and OpenCL: a case study using modern C++ libraries. SIAM J. Sci. Comput. 2013;35:C453–72.

77. Klöckner A, Pinto N, Lee Y, Catanzaro B, Ivanov P, Fasih A. PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. Parallel Comput. 2012;38:157–74.

78. MathWorks. MathWorks GPU Computing. Available from: http://nl.mathworks.com/discovery/matlab-gpu.html

79. Hung C-L, Lin Y-S, Lin C-Y, Chung Y-C, Chung Y-F. CUDA ClustalW: An efficient parallel algorithm for progressive multiple sequence alignment on Multi-GPUs. Comput. Biol. Chem. 2015;58:62–8.

80. Liu Y, Wirawan A, Schmidt B. CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions. BMC Bioinformatics. 2013;14:117.

81. Korpar M, Šikic M. SW#-GPU-enabled exact alignments on genome scale. Bioinformatics. 2013;29:2494–5.

82. García-Calvo R, Guisado J, Diaz-del-Rio F, Córdoba A, Jiménez-Morales F. Graphics Processing Unit–enhanced genetic algorithms for solving the temporal dynamics of gene regulatory networks. Evol. Bioinforma. 2018;14:11.

83. Alagic S. Relational database technology. ISBN 038796276X, 9780387962764 . 2012.

84. Gessert F, Wingerath W, Friedrich S, Ritter N. NoSQL database systems: a survey and decision guidance. Comput. Sci. 2017;32:353–65.

85. Taylor RC, Baker M, Sansom C, Stein L, Schatz M, Langmead B, et al. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC Bioinformatics. 2010;11 Suppl 1:S1.

86. Miller JJ. Graph database applications and concepts with Neo4j. Proc. South. Assoc. Inf. Syst. Conf. 2013.

2 Fast selection of miRNA candidates based on large-scale pre-computed MFE sets of randomized sequences

Published as S. Warris, S. Boymans, I. Muiser, M. Noback, W. Krijnen, J.P. Nap, “Fast selection of miRNA candidates based on large-scale pre-computed MFE sets of randomized sequences”, *BMC Research Notes*, 2014, 7:34, https://doi.org/10.1186/1756-0500-7-34.

2.1 Abstract

Background

Small RNAs are important regulators of genome function, yet their prediction in genomes is still a major computational challenge. Statistical analyses of pre-miRNA sequences indicated that their 2D structure tends to have a minimal free energy (MFE) significantly lower than MFE values of equivalently randomized sequences with the same nucleotide composition, in contrast to other classes of non-coding RNA. The computation of many MFEs is, however, too intensive to allow for genome-wide screenings.

Results

Using a local grid infrastructure, MFE distributions of random sequences were pre-calculated on a large scale. These distributions follow a normal distribution and can be used to determine the MFE distribution for any given sequence composition by interpolation. It allows on-the-fly calculation of the normal distribution for any candidate sequence composition.

Conclusions

The speedup achieved makes genome-wide screening with this characteristic of a pre-miRNA sequence practical. Although this particular property alone will not be able to distinguish miRNAs from other sequences sufficiently discriminative, the MFE-based P-value should be added to the parameters of choice to be included in the selection of potential miRNA candidates for experimental verification.

2.2 Background

Small RNAs are important molecules in the regulation of gene expression. Several classes of distinct small RNA molecules play vital roles in development, health and disease, as well as in many other biological pathways [1–4]. A particular class of regulatory small RNAs are the microRNA (miRNA) molecules. A miRNA is a ~20-23 nucleotide (nt) short, non-protein coding RNA. Together with several protein components, miRNAs reduce the amount of a target mRNA by physical interaction to notably the 3'-untranslated region (3'-UTR) of the mRNA, resulting in either degradation of that mRNA, or arrest of translation [4–6]. In rare cases, miRNAs can also upregulate expression [4, 7]. Well over 25 thousand miRNAs (miRBase Release 19; August 2012 [8]) have now been identified in many different species [9,10].

The overall biogenesis of miRNAs is well established [4,11], although details are still being discovered. In all cases except for intronic miRNAs [12], the miRNA is synthesized as a longer primary transcript known as primary miRNA (pri-miRNA), that is processed in the nucleus by the RNAse Drosha in animals and Dicer-like 1 in plants, to generate a precursor miRNA (pre-

miRNA) of about 80-100 nt in animals, 60-300 nt in plants or 60-120 nt in (animal) viruses. The pre-miRNA sequence has degenerated palindromic sequence with the characteristic secondary structure of a stem-loop hairpin. The final verdict on the total number of miRNAs in a given genome is not out yet. The total count in Release 19 of miRBase is 25,141 for all organisms and many more miRNAs are described in the primary literature. Whereas the search for miRNAs in model genomes such as human or Arabidopsis will approach saturation, identification of the full miRNA complement in other genomes is still a challenge.

As mature miRNAs are only ~22 nt in length, straight-forward alignment-based heuristic methods such as BLAST are less suitable for identifying miRNAs and their targets in a given genome or transcriptome [5,13]. The identification of miRNAs and their targets is therefore a challenge for computational pattern recognition [11,14]. Computational methods for miRNA identification focus on the typical extended stem-loop hairpin structure of the pre-miRNA, which is characterized by helical base pairing with a few internal bulges in the stem. To identify the stem-loop miRNA precursor structure from a given sequence, RNA folding programs are used, such as mfold [15], its update UNAfold [16], or RNAfold (also known as the Vienna package [17, 18], to establish the minimal free energy (MFE) of the stem-loop structure.

Increasingly sophisticated computational approaches have been proposed for the identification of pre-miRNAs, the mature miRNA sequence and its presumed target(s) [19–21], many of which are available online [22]. Many approaches are based on supposed or derived characteristics of miRNA sequences or combinations thereof [23–26]. Although all miRNAs are thought to have such properties in common, not a single property individually seems able to distinguish miRNAs sufficiently accurately from other RNA molecules with sufficient accuracy [27]. Several approaches therefore include evolutionary conservation of miRNA sequences between different species [1,28]. In these evolution-based strategies, species-specific and non -or less- conserved pre-miRNA molecules are likely to escape identification. Overall, methods available tend to show relatively high rates of false positives [22] and are possibly hampered by the use of inappropriate controls [29]. They generally result in lists too long to be feasible for experimental validation.

We here revisit a selective criterion proposed earlier, but largely unexplored because of computational costs. Statistical analyses of pre-miRNA hairpins indicated that such hairpins tend to have MFE values which are significantly lower than the MFE values based on randomized sequences with the same length and nucleotide composition, in contrast to other classes of RNA, such as transfer RNA, ribosomal RNA and messenger RNA [30–32]. In MFE analysis, the sequence composition of each candidate sequence is randomized and the MFE value based on the candidate is compared to the MFE distribution based on the randomized sequences. These data are used to calculate the probability that the MFE of the candidate is sufficiently small compared to randomized sequences [30]. This probability is here coined the empirical P-value (P_E). This P_E

establishes a useful discriminating criterion for pre-miRNA identification. It is implemented in the MiPred prediction tool [33], that helps to decide for a single sequence whether it is a pre-miRNA hairpin. However, the computation of large numbers of MFE values per candidate sequence to be able to calculate P_E is computationally demanding, which precludes application to genome-wide analyses. Solutions proposed in the literature are a probabilistic implementation of the MFE computation [34] or asymptotic Z-scores of the MFE distribution based on precomputed tables [35]. We here present a novel approach that requires the computation of only the MFE based on the candidate sequence. This approach enables the routine evaluation of potential miRNA structures on a genome-wide scale that could be integrated as part of an existing approach for processing potential miRNA sequences [20, 36].

2.3 Materials and methods

2.3.1 Data

The miRNA data set was downloaded from the miRbase repository [8–10] (releases 9.2 and 15), consisting of 4,584 and 15,172 pre-miRNA sequences respectively. The genomic sequence of the Epstein Barr virus type 1 [37] was downloaded from Genbank NCBI [gi|82503188|ref|NC_007605.1]. The test set with 250,000 random sequences was generated with a small C program.

2.3.2 Hardware

Computations were performed on a 200+-node Debian Linux-based network. A dedicated server is running Network File System (NFS)-based software for file management and Condor software ([38]; version 7.6.1) for grid management [39].

2.3.3 RNA folding software

The minimal free energy of a sequence was computed with a local implementation of the Hybrid software (version 2.5) of the UNAFold software package [16,40]. UNAFold extends and replaces the earlier mfold application [15]. The software was adjusted to enhance the performance about three-fold by optimizing computation-intensive computational steps without changing the underlying algorithm. All RNA molecules were folded as single strands at 30°C, a sodium concentration of 1.0 M and the option –E (energy only, no plots). In case of sequences that are not able to fold properly, the Hybrid software assigns an MFE of $+\infty$.

2.3.4 Randomization and visualization

Distribution fitting, P-values and other statistics were computed with the software suite R (version 2.7.1.) [41]. To randomize sequences while maintaining the nucleotide composition, the Fisher-Yates shuffling procedure for selection without replacement was implemented in C, with

appropriate unbiased randomization [42]. For any candidate sequence, the empirical P-value P_E was computed as $P_E = X/(N+1)$ [30], where X is the number of sequences with an MFE lower than or equal to the MFE based on the candidate sequence and N is the number of randomized sequences considered. In this study, N is taken as 1,000, in correspondence with an earlier study [30]. As a consequence, the lower bound of P_E is zero (for X = 0) and the next lowest value is 0.000999 (for X=1). There are no additional assumptions necessary with respect to the shape of the distribution of the MFE values [30]. The computed MFE values based on the randomized sequences were transformed into a normal (Gaussian) distribution defined by the mean and standard deviation of the MFE values. The normal distribution-derived P_N of the MFE based on the given candidate sequence is being computed using the mean and standard deviation of that distribution. Results were visualized with R and MatLab (release 13).

2.3.5 Multidimensional interpolation

A database of entry RNA sequences, with a length of 50 to 300 nt and a step size of 5 nt, was generated by computer. This range covers the length of most known pre-miRNAs, except for some plant miRNAs [43]. For each sequence length, the nucleotide composition of the sequence was varied in such a way that each of the four nucleotides occurs at least once (for sequences < 100 nt) or at least at 1% (for sequences > 100 nt). Per sequence length, individual sequences were generated with a step size for an individual nucleotide of 2%, except in the range from 20-70% for an individual nucleotide where a step size of 1% was used. The procedure in numbers is as follows: for a population of sequences with a length of 50nt, the first nucleotide composition consists of 1% A, 1% U, 1% C and 97%G. In the next step the composition is 2% A, 1% U, 1% C and 96% G and so on. Then the length is increased to 55 nt and the procedure is repeated for the nucleotide composition, etc. This procedure generated a set of 1.4×10^6 entry sequences. For each of the individual entry sequences, a sequence set of thousand randomized shuffles was generated by Fisher-Yates randomization [42]. This procedure represents a selection without replacement, therefore maintains the nucleotide composition (mononucleotide shuffling). Sequence sets in which one or more shuffled sequences had an MFE of $+\infty$ were discarded and only the sequence sets with 1,000 MFE values were considered to maintain statistical validity. This way, a total of 1.05×10^6 sequence sets were generated. For each population, the mean MFE and standard deviation were computed and stored in a MySQL database together with the sequence composition in absolute nucleotide counts. To calculate the mean and standard deviation for any candidate sequence, an interpolation algorithm was implemented in C++ using sparse matrix data management for optimal memory use [44]. A sparse matrix contains only the values of interest and all zero or unknown values are not stored. The resulting data structure contains only the nucleotide composition analyzed and not all possible compositions, therefore the data can be stored in memory and searched efficiently. The Hybrid software used for RNA folding [16] was

integrated within this application to enhance performance.

2.3.6 Sliding window analysis

To analyze whole genomes for the presence of potential pre-miRNA candidates using the pre-computed MFE data outlined above, a sliding window approach was implemented in C++. The smallest window length was set at 50 nt, incremented with a step size of 10 nt to a maximum of 300 nt. For each window length, the step size for sliding was set at 10% of the window length. For each window, the MFE was computed and the nucleotide composition of the sequence was determined. Based on the sequence composition, the appropriate mean and standard deviation were estimated by interpolation (see Results) using the data search space generated. The normal distribution function was used to calculate P_N of the MFE of the window.

2.4 Results

In the identification of potential pre-miRNA candidates in genomic sequences, the MFE based on the sequence relative to the distribution of random sequences with the same nucleotide composition is a potentially valuable criterion. However, the estimation of the empirical P_E as parameter for the distance between the MFE based on a candidate sequence and the MFEs of randomized sequences is computationally intensive. It requires the computation of the MFE for all randomized sequences. To use the MFE distribution as criterion more comfortably, computations should be considerably faster. We here show the feasibility of the use of the normal distribution for the computation of P_N as approximation of P_E and for the interpolation of the distribution for any given sequence with the help of pre-computed MFE distributions of random sequences.

2.4.1 Pre-computed MFE distributions of random sequences

A total of 1.4×10^6 entry sequences covering the length classes representative for most known pre-miRNA (50 – 300nt), were generated. Each entry sequence was shuffled 1,000 times and based on each of the generated sequences the MFE was calculated giving a total of 1,053,248 populations, each consisting of 1,000 random sequences. In 346,752 generated populations one or more random sequence could not fold properly. When the hybrid software is not able to give a stable structure, the random sequence is considered not to fold properly and is therefore not included because it skews the data. For a single sequence on a standard desktop PC, the MFE computation by the Hybrid software requires approximately 0.2 sec CPU time. The $1.4 \times 10^6 \times 1{,}000$ computations would therefore have taken about 8.8 CPU years on a standard PC. Using idle CPU cycles on our grid, it took about 2 months grid time to complete all computations. All MFE values were computed for an annealing temperature of 30°C, but as MFE values and distributions change in a linear way with temperature (results not shown), the approach presented and data generated are, if so desired, suitable for, or comparable with, other folding temperatures.

Randomized sequence sets can reasonably be considered to reflect a normal distribution. The examples for sets with 25% nucleotide composition are shown in Figure 2.1. Other compositions give similar results (data not shown). Such a normal distribution was demonstrated earlier for randomized sequences [45], although the distribution may not be an exact Gaussian distribution [34]. The MFE data of random sequences are therefore suitable for deriving the normal estimate P_N of P_E, based on mean, standard deviation and the normal distribution function. This way, P_N is equivalent to the Z-score of the MFE, defined as the number of standard deviations by which the MFE based on a candidate sequence deviates from the mean MFE of the set of shuffled sequences [31,45].

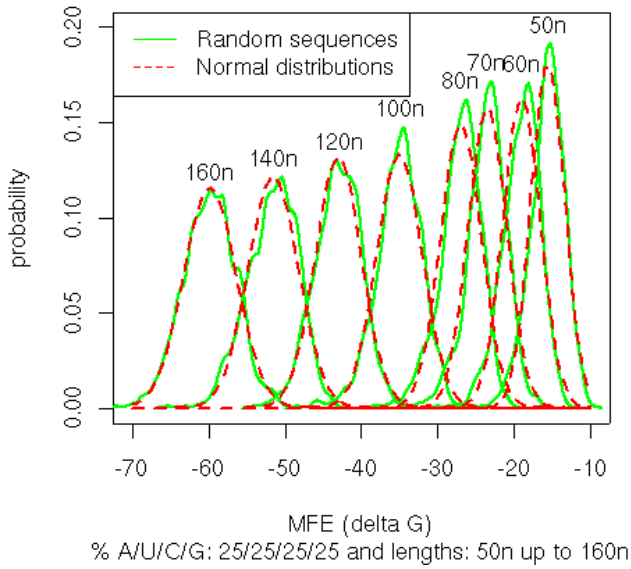

Figure 2.1. Distribution of P_E and P_N of sequences of different lengths. For a candidate sequence with the given length in nucleotides n (50 to 160) and a composition of 25% of each nucleotide (AUGC), the MFE of 1000 randomized sequences was calculated. The distribution was computed and plotted (green) using the distribution density function in R. The average mean and standard deviation of the resulting MFE sequence set was used to define the normal distribution function (red). The good correspondence between the two distributions shows that the normal distribution-based probability P_N is a good approximation for the empirical probability P_E.

For each sequence set, the mean and standard deviation was stored in a database together with the sequence composition. An example of the distribution of the mean MFE value of all sequence sets of 100 nt in length with different sequence compositions is shown in Figure 2.2. The 3D contour plot shows that the sets of sequences with high percentages of C and G nucleotides have low mean MFE values, which reflects the higher energy in C-G pairing. RNA molecules with an abundance of for example A and G are much less likely to form a stable structure and the set of 1,000 random sequences will therefore have a high mean MFE. The plot shows that the mean MFE values decrease in an almost linear fashion from the low values for sequences with high C and G compositions to the outer edges.

2.4.2 Multidimensional interpolation for candidate sequences

For the on-the-fly computation of the MFE distribution based on a given candidate sequence, the pre-computed data are used for multidimensional interpolation. For each candidate sequence, the composition of the sequence is determined by counting nucleotides. Sequence compositions with a squared Euclidean distance up to 5 in the surrounding search space are identified: the length of the sequence is therefore not taken into account. This value was selected on the basis of the analysis of known miRNAs. These analyses showed this threshold gives the smallest difference in P-value (data not shown) when comparing the P_N to the P_E of mirbase entries. For sequence compositions that have no points within this distance no prediction can be made and a P_N of 1.0 is given. From the selected near-by sequences, the mean and standard deviations are retrieved from the database. For the candidate sequence, both mean and standard deviation of the MFE distribution are determined by interpolation using the data from the nearby sequence compositions, weighted based on their Euclidean distance to the candidate sequence. The sum of the weighted values gives the estimated mean and standard deviation of the MFE distribution based on the randomized sequences from the candidate sequence. Mathematically, the formulae to derive the estimated average μ_e are expressed as:

$$\bar{\mu}_e = \frac{\sum_{i=0}^{N-1} \omega_i \mu_i}{\sum_{i=0}^{N-1} \omega_i} \text{ with } \omega_i = \frac{d_t - d_i}{d_t}, \; d_i = \sqrt{\sum_{i=0}^{3} (x_i - y_i)^2} \text{ and } d_t = \sum_{i=0}^{N-1} d_i$$

where w_i is the weight per data point based on Euclidean distance, d_i is the distance per point, for which x_i and y_i are the nucleotide counts of the sequence in the search space and d_t is the total distance over N points. N varies per candidate sequence. Even in the case were the distance to a point is zero ($d_i = 0$), more points are used to estimate the population average. Testing on sequences with the same compositions during computations would slow the software down and this situation is unlikely therefore it was not included in the software.

The use of P_N as normal approximation of P_E was evaluated by comparing both probabilities for different sequences. In Figure 2.1, the comparison between many P_E (green) and P_N (red) is shown

for a range of sequences with different lengths but the same nucleotide compositions. Other compositions give similar results (data not shown). The excellent goodness-of-fit demonstrates the suitability of the normal approximation P_N as criterion for the evaluation of pre-miRNA candidate sequences.

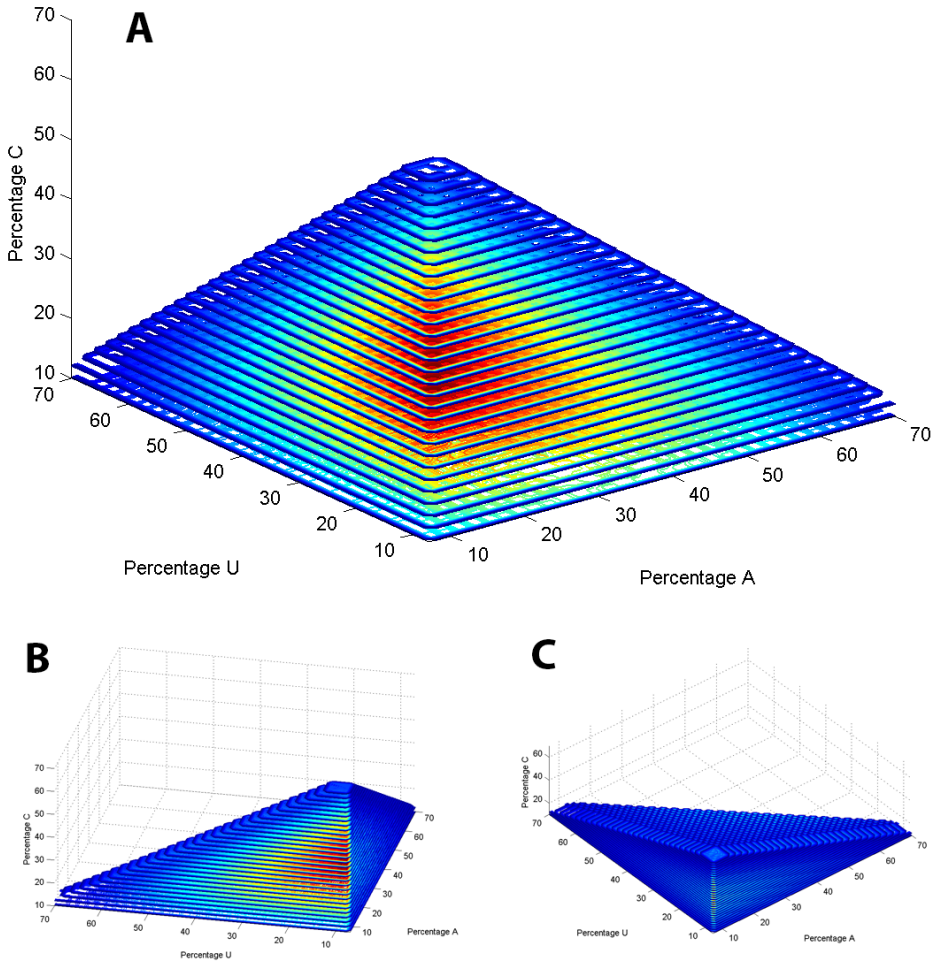

Figure 2.2. Mean MFE distribution of sequences 100 nt in length. (A) The mean MFE of 1000 sequences with the indicated composition is plotted in a 3D contour plot (Matlab) with the percentage of three nucleotides in the sequence specified on the three axes. The false color scale indicates a relative measure of the mean MFE: red a relatively low MFE value with a ΔG (Gibbs free energy change) ≤ -80 kcal/mol, yellow an intermediate MFE value (-80 kcal/mol < ΔG ≤ -40 kcal/mol) and blue a relatively high MFE value (ΔG > -40 kcal/mol). (B,C) The same distribution as in (A) is shown at two different angles to help interpretation and to prevent optical illusions.

The estimation of the standard deviation of the MFE distribution based on the candidate sequence is based on the approach for estimating the mean as described in the previous section. The mean and standard deviation uniquely define the normal distribution function of the candidate sequence. With the two values, P_N is computed as the normal probability of MFE values smaller than the MFE based on the candidate sequence. This way, for each candidate sequence, only the MFE of the structure based on this sequence needs to be computed, speeding up computations approximately a thousand-fold when thousand shuffled sequences are used. As the calculations of the estimated mean, standard deviation and the P-value based on this normal distribution take time as well, the software is at least several hundred times faster for short sequences and faster for long candidates: the Hybrid software used is slower for longer sequences, as there are more secondary structures. Calculating the structures of 1000 candidates takes therefore considerably more time than the estimation process. Based on the running time of an example run it would take over 260 seconds to calculate 1000 MFE values. The calculation of P_N takes approximate a second, including the calculation of the MFE.

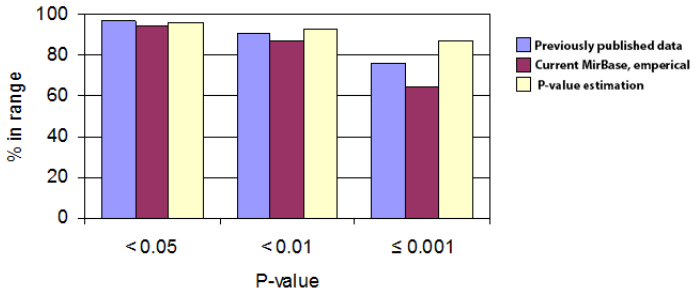

Figure 2.3. Relative performance of MFE-based P-value estimations. The percentage of pre-miRNAs with a P-value smaller than indicated is plotted for data previously published [30], newly computed values from release 9.2 of MirBase based on the same method and computed based on the interpolation method developed here. The previously published percentages based on P_E were 97%, 91% and 76%, respectively, whereas based on the release 9.2 it is 95%, 87% and 65%, and P_N 96%, 93% and 87%, respectively.

We evaluated the performance of P_N compared to P_E for selection with respect to the entries in the MiRBase registry. In Figure 2.3, the distribution of P_E over the pre-miRNA molecules as published previously [30] is compared with the P_E computed of the current pre-miRNAs from miRbase with the same method [30] and the P_N as estimated by interpolation. It shows that the interpolation approach performs well. The difference between P_E and P_N reflects that the P_N distribution is continuous, whereas with 1,000 randomizations, the P_E distribution is discrete with a step size of 1/1001 = 0.000999. Although also P_N is estimated on the basis of 1,000

randomizations, the continuity of the distribution allows more strict settings for P_N and allows a more sensitive ranking in the lower P-value ranges. In Table 2.1, the percentages of pre-miRNAs that conform to given settings of the P-value are shown. With a threshold of $P_N = 1x10^{-4}$, about 66.8% of all known pre-miRNAs are characterized by an MFE value well outside the distribution of MFE values of randomized sequences. Only 1% could not be estimated: the sequence was either too long or had no populations in the data set within the given distance.

To verify the performance of the applications on random sequences, a test set of 250,000 sequences was generated; all of different lengths and composition. The MFE and P_N were calculated for each of these sequences. The calculations were finished in 20 minutes. In Table 2.1 the results are shown. Of these random sequences, 4% had no stable structure and hence no MFE and 17% had no populations in the data set within the given distance. Of the remaining sequences only 3% had a $P_N < 0.05$.

Table 2.1. Percentage of sequences with low P-values.

miRNA data set	Total # sequences analyzed	% sequences with an error / not found	% sequences with P <0.05	% sequences with P <0.01	% sequences with P <=0.001	% sequences with P <=0.0001
Random	250000	21.3	3.0	1.5	0.8	0.5
MirBase	15172	1.0	89.9	84.4	75.9	66.8
EBV known	25	0.0	96.0	92.0	88.0	76.0
EBV genome	566988	0.1	19.1	10.7	5.7	3.6

This table shows the percentage of sequences below the given P-value in four different data sets. A data set of random sequences shows a very low percentage with a P-value <= 0.001. A high percentage of known miRNAs sequences is within the same range.

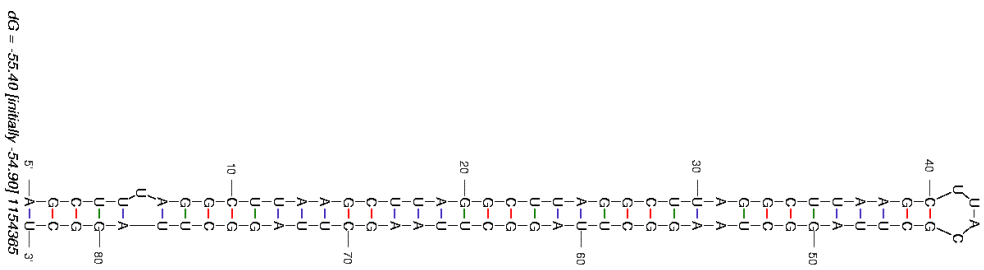

Figure 2.4. Hairpin with internal repeat structure. Example of an RNA sequence with a large difference in P_N versus P_E, depending on the method of randomization (see text for details).

2.4.3 Shuffling inconsistencies

In the analyses above, a mononucleotide randomization method (Fisher-Yates algorithm) was used [42], whereas in the literature a dinucleotide randomization method was recommended and used [30, 31, 46]. In genome-wide analyses of candidate sequences based on P_N, we observed that several candidate sequences behaved oddly: whereas dinucleotide shuffling yielded P_N = 0,98, reflecting not a likely candidate, mononucleotide shuffling as performed here resulted in a P_N<< 0.001, indicating a possible candidate. An example of such a sequence is given in Figure 2.4, which shows a predicted hairpin structure for this sequence. To prevent such sequences interfering with the analysis of the distribution, the mononucleotide randomization method was used.

2.4.4 Added value for miRNA prediction

Having validated the P_N interpolation method with known miRNAs, we now show the added value for whole genome screening. We have evaluated a small viral genome for putative pre-miRNAs regions. According to miRBase (both in release 15 and 19), this viral genome has 25 known pre-miRNA sequences. Of the 25 known Epstein-Barr virus (EBV) miRNAs, 24 have a P_N < 0.05 with 22 having a P_N <= 0.001 (Table 2.1). Both strands of the dsDNA genome sequence of the human Epstein-Barr virus type 1 [37] were converted into RNA and investigated for potential pre-miRNA sequence. For each of the total 566,988 windows of length 50-230 nt, the MFE and the P_N were computed. In contrast to the test set with random sequences, in EBV only 0.05% of the windows could not be estimated due to no sequence compositions within the given distance or because the sequence did not have a stable 2D structure. This shows that the application performs very well for viral genomic DNA. The percentage of windows with a P_N <=0.001 is higher than in the test set with random sequences (Table 2.1). There is also the effect of overlapping windows: a pre-miRNA of length 150 will be found in several windows of length 200. Many of these candidate windows are in repeat regions (Figure 2.5). These windows can be discarded as not being viable locations for miRNAs: there are no known miRNAs within the EBV repeat regions. Although current research shows that in some organisms miRNAs can be found in repeat regions [47], we suggest inspection of other regions first to limit the number of relevant candidate regions.

To visualize regions of interest, the windows with a P_N <= 0.001 are placed in a separate data set and marked with a value of one. The windows of different lengths are then combined in the Integrated Genome Browser [48] (Figure 2.5). With 19 different window lengths, the maximum of the resulting graph is 19. This indicates that all windows covering this location have a P_N <=0.001. The peaks in the graph show regions of interest which require further research. The graph shows 18 regions-of-interest in the plus strand where 9 or more windows have P_N <= 0.001. Using these selection criteria, the regions of 12 known miRNAs are found (Figure 2.5). Using less than 9 windows will give more regions-of-interest and will also show more known miRNA

regions, but will introduce more false positives as well. The analysis of the EBV genome shows that a (small) whole genome screening using P_N-estimation results in a limited number of regions-of-interest for further investigation.

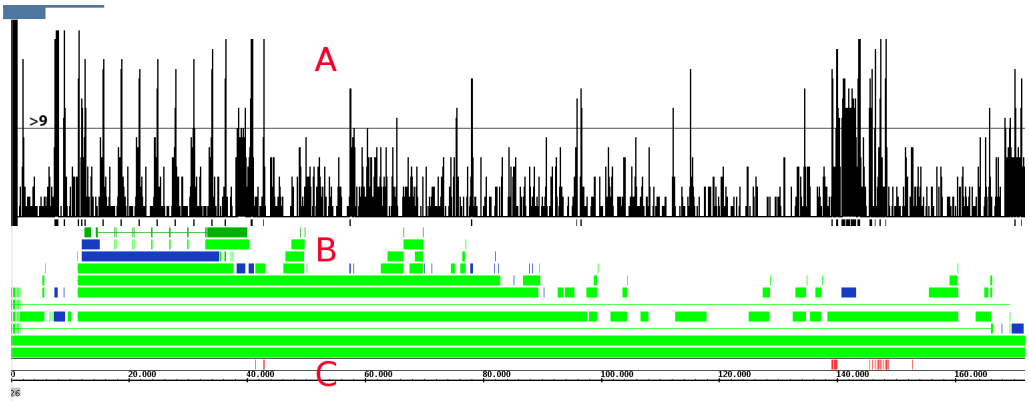

Figure 2.5. Identification of potential pre-miRNA candidates in the Epstein-Barr virus genome sequence. The genomic sequence is shown on the x-axis. The upper track (red A) shows the amount of windows covering the particular region that have a $P_N <= 0.001$. A distinct peak gives a region of interest for a candidate miRNA. By discarding peaks within a repeat region (here shown in blue) and selecting peaks at or above 9 hits, 18 new regions of interest are found (plus strand). Also, 12 known miRNA are found. The green bars indicated by the red B show the EBV genome annotation (gi|82503188|ref|NC_007605.1). The lower part of the graph (red C) shows the EBV genome locations with the red bars indicating locations of the known miRNAs.

2.5 Conclusions

Previous research has indicated that the MFE based on a miRNA sequence is significantly lower than the MFE based on shuffled sequences with the same composition, in contrast to the MFE of other non-coding RNAs [30,31,49]. As the computation of an MFE is demanding, this characteristic of miRNAs precludes genome screening of sequences for their MFE distribution. With thousand randomizations per candidate sequence, the genome-wide screening of a million (10^6) candidates would require a billion (10^9) computations. These would take well over six year to finish on a current standard desktop computer.

We have presented a method to speed up analyses of the MFE distribution considerably, based on the normal approximation of pre-calculated MFE distributions based on random sequences, combined with a fast implementation of a multidimensional interpolation of distributions in sequence space. The data cover the search space for all RNA molecules with a length from 50 to 300 nt, in total roughly equaling the sum over 4^i for $i = 50 \ldots 300 \approx 5.5 \times 10^{180}$ sequences. With three data points per sequence (mean, standard deviation and composition), this would generate an immense database, whereas the resulting data space here established is based on 1.1×10^6

sequences and takes about 30 Mb. The latter is easily handled by standard amounts of RAM. The results show that although the newer miRNAs added to miRBase since 2006 seems to comply somewhat less with this criterion than the miRNAs analyzed before [50], the new approach developed here performs well on known pre-miRNAs (Figure 2.3).

Sequence sets of 1,000 with at least one non-folding member were discarded. Yet, it could be argued that higher accuracy would be gained with more sets. The data is well distributed over the sequence data space (Figure 2.2). The interpolation of MFE distributions is based on a threshold of the Euclidian distance of the surrounding data points. This implies that for different candidate sequences different amounts of pre-computed data are used to estimate the MFE distribution. This prevents interpolation issues at the boundaries of the data space where less points are available. The data reduction and interpolation results in considerably faster computation of the likelihood that the MFE of the sequence is markedly lower than equivalent randomized sequences. This obviates the need for on-the-fly computation of the MFE values based on the randomized sequences.

The particular type and number of randomizations is an issue. Whereas it was thought to be important to maintain not only the mononucleotide compositions, but also the dinucleotide distribution [46], the results shown for the behavior of miRNAs in either way of shuffling [30, 51] indicate no relevant difference, or even a slightly better performance of mononucleotide shuffling. These findings indicate that for miRNA prediction dinucleotide shuffling is not more optimal than mononucleotide shuffling. This is in agreement with the demonstration that all base pairings in an RNA molecule should be taken into account [52]. There is, in addition, uncertainty over the quality of the dinucleotide shuffling algorithm [35,51]. As demonstrated here, particular sequences behave oddly with respect to dinucleotide shuffling (Figure 2.4) and may distort the distribution derived from the computed MFE values. Inspection of such sequences indicated that these sequences contain a particular combination of repeat units in such a way that the dinucleotide shuffling is not changing the sequence in terms of MFE distribution. As a result, the candidate MFE is part of the distribution of shuffled sequences. As Fisher-Yates shuffling is the most random, this would seem to be the better method. We have followed the earlier recommendation of performing at least 1,000 randomizations [30], whereas other investigations use 500 [31] or 10,000 [51]. As few as 100 randomizations were recommended as sufficient to establish a reasonable Gaussian distribution [45].

In view of the gain in computing speed accomplished with P_N, it has become feasible to consider genome-wide screenings for pre-miRNA candidates based on P_N. The analyses here presented for the relatively small Epstein Barr virus demonstrate that indeed such analysis is now within reach. For a human genome, however, the approach will still ask a considerable computational effort. Moreover, the MFE alone is not able to distinguish miRNAs from other sequences sufficiently

discriminative: it has to be integrated with other parameters. The P_N approach presented here can therefore be better implemented as part of, or next to, other approaches [20,22]. Such approach would generate added value for such miRNA identification algorithms or pipelines. The application of this criterion will add to enhanced selectivity of miRNA discovery pipelines and help to limit the number of candidates for experimental validation and confirmation.

The advent of high throughput DNA sequencing technologies were shown to be particularly suitable for the analyses of the small RNA complement of RNA populations [4]. The identification of true miRNAs in such data sets is still a challenge to which the P_N analysis may contribute. The possibility of a one-time effort to pre-compute sequence parameters that will facilitate future analyses should be considered an approach that could generate considerable added value for larger grid environments in future bioinformatics.

2.6 Acknowledgements

We would like to thank Piet Plomp, MSc and Marcel Kempenaar, BASc for help with the computer grid, Dr. Peter Terpstra (Department of Genetics, University Medical Center Groningen, The Netherlands) and Rudi van Bavel BASc (now at Keygene NV) for sharing ideas and Dr. Mark Fiers at Plant Research International, Wageningen University and Research Centre, Wageningen, The Netherlands (now at Plant & Food Res. Ltd, New Zealand), for valuable discussion and input. This research was funded in part by a PhD fellowship (Sven Warris) from Hanze University of Applied Sciences Groningen.

2.7 References

1. Chapman EJ, Carrington JC: Specialization and evolution of endogenous small RNA pathways. Nat. Rev. Genet. 2007, 8:884–896.
2. Almeida MI, Reis RM, Calin GA: MicroRNA history: Discovery, recent applications, and next frontiers. Mutat. Res. 2011.
3. Abbott AL: Uncovering new functions for microRNAs in *Caenorhabditis elegans*. Curr. Biol. 2011, 21:R668–71.
4. Pasquinelli AE: MicroRNAs and their targets: recognition, regulation and an emerging reciprocal relationship. Nat. Rev. Genet. 2012, 13:271–82.
5. Bartel DP: MicroRNAs: Target recognition and regulatory functions. Cell 2009, 136:215–233.
6. Bartel DP: MicroRNAs: genomics, biogenesis, mechanism, and function. Cell 2004, 116:281–297.
7. Vasudevan S, Tong Y, Steitz JA: Switching from repression to activation: MicroRNAs can up-regulate translation. Science. 2007, 318:1931–1934.
8. MirBase [http://www.mirbase.org].
9. Griffiths-Jones S, Saini HK, van Dongen S, Enright AJ: miRBase: tools for microRNA genomics. Nucleic Acids Res. 2008, 36:D154–D158.
10. Kozomara A, Griffiths-Jones S: miRBase: integrating microRNA annotation and deep-sequencing data. Nucleic Acids Res. 2011, 39:D152–7.
11. Ghosh Z, Chakrabarti J, Mallick B: miRNomics-The bioinformatics of microRNA genes. Biochem. Biophys. Res. Commun. 2007, 363:6–11.
12. Westholm JO, Lai EC: Mirtrons: microRNA biogenesis via splicing. Biochimie 2011, 93:1897–904.

13. Freyhult EK, Bollback JP, Gardner PP: Exploring genomic dark matter: A critical assessment of the performance of homology search methods on noncoding RNA. Genome Res. 2007, 17:117–125.

14. Lindow M, Gorodkin J: Principles and limitations of computational microRNA gene and target finding. DNA Cell Biol. 2007, 26:339–351.

15. Zuker M: Mfold web server for nucleic acid folding and hybridization prediction. Nucleic Acids Res. 2003, 31:3406–3415.

16. Markham NR, Zuker M: DINAMelt web server for nucleic acid melting prediction. Nucleic Acids Res. 2005, 33:W577–W581.

17. Hofacker IL: Vienna RNA secondary structure server. Nucleic Acids Res. 2003, 31:3429–3431.

18. Gorodkin J, Hofacker IL: From structure prediction to genomic screens for novel non-coding RNAs. PLoS Comput. Biol. 2011, 7:e1002100.

19. Oulas A, Karathanasis N, Louloupi A, Poirazi P: Finding cancer-associated miRNAs: methods and tools. Mol. Biotechnol. 2011, 49:97–107.

20. Sarker R, Bandyopadhyay S, Maulik U: An overview of computational approaches for prediction of miRNA genes and their Targets. Curr. Bioinform. 2011, 6:15.

21. Krzyzanowski PM, Muro EM, Andrade-Navarro MA: Computational approaches to discovering noncoding RNA. RNA 2012, 3:567–79.

22. Tan Gana NH, Victoriano AFB, Okamoto T: Evaluation of online miRNA resources for biomedical applications. Genes Cells 2012, 17:11–27.

23. Zheng Y, Hsu W, Lee ML, Wong L: Exploring essential attributes for detecting MicroRNA Precursors from background sequences. Lect. Notes Bioinforma. 2006, 4316:131–145.

24. Van der Burgt A, Fiers MWJE, Nap J-P, van Ham RCHJ: *In silico* miRNA prediction in metazoan genomes: balancing between sensitivity and specificity. BMC Genomics 2009, 10:204.

25. Tempel S, Tahi F: A fast *ab-initio* method for predicting miRNA precursors in genomes. Nucleic Acids Res. 2012, 40:e80.

26. Liu X, He S, Skogerbø G, Gong F, Chen R: Integrated sequence-structure motifs suffice to identify microRNA precursors. PLoS One 2012, 7:e32797.

27. Bentwich I: Identifying human microRNAs. Curr. Top. Microbiol. Immunol. 2008, 320:257–269.

28. Lindow M, Jacobsen A, Nygaard S, Mang Y, Krogh A: Intragenomic matching reveals a huge potential for miRNA-mediated regulation in plants. PLoS Comput Biol 2007, 3:e238.

29. Ritchie W, Gao D, Rasko JEJ: Defining and providing robust controls for microRNA prediction. Bioinformatics 2012, 28:1058–61.

30. Bonnet E, Wuyts J, Rouze P, Van de Peer Y: Evidence that microRNA precursors, unlike other non-coding RNAs, have lower folding free energies than random sequences. Bioinformatics 2004, 20:2911–2917.

31. Freyhult E, Gardner PP, Moulton V: A comparison of RNA folding measures. BMC Bioinformatics 2005, 6:241.

32. Ng KLS, Mishra SK: *De novo* SVM classification of precursor microRNAs from genomic pseudo hairpins using global and intrinsic folding measures. Bioinformatics 2007, 23:1321–1330.

33. Jiang P, Wu H, Wang W, Ma W, Sun X, Lu Z: MiPred: classification of real and pseudo microRNA precursors using random forest prediction model with combined features. Nucleic Acids Res. 2007, 35:W339–W344.

34. Rivas E, Eddy SR: Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. Bioinformatics 2000, 16:583–605.

35. Clote P, Ferre F, Kranakis E, Krizanc D: Structural RNA has lower folding energy than random RNA of the same dinucleotide frequency. RNA 2005, 11:578–591.

36. Gomes CPC, Cho J-H, Hood L, Franco OL, Pereira RW, Wang K: A review of computational tools in microRNA discovery. Front. Genet. 2013, 4:81.

37. De Jesus O, Smith PR, Spender LC, Karstegl CE, Niller HH, Huang D, Farrell PJ: Updated Epstein-Barr virus (EBV) DNA sequence and analysis of a promoter for the BART (CST, BARFO) RNAs of EBV. J. Gen. Virol. 2003, 84:1443–1450.

38. Condor software website [http://www.cs.wisc.edu/condor].

39. Thain D, Tannenbaum T, Livny M: Distributed computing in practice: The Condor experience. Conc Comp Pr. Exp 2005, 17:323–356.

40. UnaFold Package [http://dinamelt.bioinfo.rpi.edu/download.php].

41. R Project [http://www.r-project.org].

42. Black PE: Fisher-Yates shuffle. Dict. algorithms data struct. U.S. National Institute of Standards and Technology; 2005.

43. Thakur V, Wanchana S, Xu M, Bruskiewich R, Quick WP, Mosig A, Zhu X-G: Characterization of statistical features for plant microRNA prediction. BMC Genomics 2011, 12:108.

44. Arnold G, Hölzl J, Köksal AS, Berkeley UC: Specifying and verifying sparse matrix codes. Discovery 2010:1–13.

45. Le SY, Maizel J V: A method for assessing the statistical significance of RNA folding. J. Theor. Biol. 1989, 138:495–510.

46. Workman C, Krogh A: No evidence that mRNAs have lower folding free energies than random sequences with the same dinucleotide distribution. Nucleic Acids Res. 1999, 27:4816–4822.

47. Zhao Y, Xu H, Yao Y, Smith LP, Kgosana L, Green J, Petherbridge L, Baigent SJ, Nair V: Critical role of the virus-encoded microRNA-155 ortholog in the induction of Marek's disease lymphomas. PLoS Pathog. 2011, 7:e1001305.

48. Integrated Genome Browser [http://bioviz.org/igb/].

49. Niu QW, Lin SS, Reyes JL, Chen KC, Wu HW, Yeh SD, Chua NH: Expression of artificial microRNAs in transgenic *Arabidopsis thaliana* confers virus resistance. Nat. Biotechnol. 2006, 24:1420–1428.

50. Bonnet E, Van De Peer Y, Rouze P: The small RNA world of plants. New Phytol. 2006, 171:451–468.

51. Ng KLS, Mishra SK: Unique folding of precursor microRNAs: quantitative evidence and implications for *de novo* identification. RNA 2007, 13:170–187.

52. Parisien M, Major F: The MC-Fold and MC-Sym pipeline infers RNA structure from sequence data. Nature 2008, 452:51–55.

3 Flexible, fast and accurate sequence alignment profiling on GPGPU with PaSWAS

Published as S. Warris, F. Yalcin, K.J.L. Jackson, J.P. Nap, “Flexible, fast and accurate sequence alignment profiling on GPGPU with PaSWAS”, *PLoS ONE* (2015), 10 (4), https://doi.org/10.1371/journal.pone.0122524

3.1 Abstract

Background

To obtain large-scale sequence alignments in a fast and flexible way is an important step in the analyses of next generation sequencing data. Applications based on the Smith-Waterman (SW) algorithm are often either not fast enough, limited to dedicated tasks or not sufficiently accurate due to statistical issues. Current SW implementations that run on graphics hardware do not report the alignment details necessary for further analysis.

Results

With the Parallel SW Alignment Software (PaSWAS) it is possible (a) to have easy access to the computational power of NVIDIA-based general purpose graphics processing units (GPGPUs) to perform high-speed sequence alignments, and (b) retrieve relevant information such as score, number of gaps and mismatches. The software reports multiple hits per alignment.

Conclusions

The added value of the new SW implementation is demonstrated with two test cases: (1) tag recovery in next generation sequence data and (2) isotype assignment within an immunoglobulin 454 sequence data set. Both cases show the usability and versatility of the new parallel Smith-Waterman implementation.

3.2 Background

Currently available next generation sequencing platforms [1] produce millions of short reads, from 30 bases up to several hundred bases, which are analyzed for SNPs [2], miRNAs [3] and other short sequences, or used for purposes such as whole genome (re)sequencing [4]. Fast, flexible and highly accurate alignment software is an important tool for analyzing such sequencing data. The alignment software should be able to process the large amounts of data within a limited timeframe, preferably on low cost and high-speed hardware. The software also needs to be highly accurate, giving the exact locations of mismatches, gaps, etc. Moreover, it is important that the application is flexible, so it can be used for many different purposes.

The Smith-Waterman (SW) algorithm is an exact method to perform local sequence alignments. The algorithm provides a dynamic programming approach of order $O(n^2)$, which makes the algorithm computationally slow [5]. BLAST [6] and related heuristic approaches [7] are used to search sequence databases as well as aligning sequences. Through seeding and other statistical methods, BLAST reduces the overall number of local alignments needed [6]. BLAST is very flexible and in most cases fast enough to perform the analyses required. For short sequences and highly accurate alignments, BLAST is however less suitable [8].

Dedicated software is used for finding single-nucleotide polymorphisms (SNPs) and other small differences between sequences. SOAP, for example, gives the user the location of SNPs using seeding and hash lookups, but is limited by the small number of SNPs allowed [9]. SOAP makes assumptions about SNP frequencies and uses statistical filters [10] which makes it, like BLAST, less accurate than a full SW alignment.

In recent years the use of graphics cards as platform for non-graphical data processing has taken off [11]. This programming platform provides ease of access to the computing power of the relatively cheap graphics processing unit (GPU). The programming language for NVIDIA GPUs is CUDA, which is an extension of C/C++. Numerous SW implementations have been presented upon the release of the first CUDA-enabled graphics cards and have shown that GPUs can deliver significant speed-ups compared to CPU implementations [12–17]. Some implementations, aimed at searching reads in large genome or protein databases, give a single location and highest score for each sequence. These implementations are, therefore, not able to indicate multiple hits and do not produce an alignment. Without the exact alignment it is for example impossible to find the exact location of a base change in a SNP. Other implementations have specific functionality such as accelerating protein BLAST [13]. GPGPU-based applications run on low cost, easily available hardware. The graphics cards fit in most standard desktop PCs as well as in high-end, high-performance servers. Compared to other dedicated hardware, the price-to-performance ratio favors GPGPU solutions. With the release of other GPGPU-based bioinformatics tools such as GPU-BLAST, the hardware can be used for other purposes, in contrast to dedicated hardware such as field-programmable gate arrays.

In this paper we present a new GPGPU-implementation of the SW algorithm that is not only fast and accurate, but which also generates detailed information about each alignment for inspection. The alignment information supplied includes the location of the hit, the number of matches, mismatches and gaps, as well as the alignment profile: the visual representation of the alignment. The implementation is dubbed Parallel Smith-Waterman Alignment Software (PaSWAS). PaSWAS can use any scoring matrix, so the application is able to align DNA, RNA or protein sequences. The implementation also allows for more than one profile per sequence alignment, which is useful when a sequence is contained in its target more than once or is split up in the target with a large segment between the parts.

To show the added value of PaSWAS, we analyzed two datasets that each presented a different scientific challenge. These examples are added to show the applicability of PaSWAS in general research settings and are not intended as benchmarks of any available software for each case. In the supporting information, we show how the results of PaSWAS compare to the BLAST-based analysis that were routinely performed at the institutes involved (3.8 Supporting information S1).

Dataset (1): identity (ID) tag recovery from 454 sequence reads. An essential part of most high-throughput sequence analysis is sequence cleaning, i.e. the removal of adaptor sequences, tags or vector contamination prior to subsequent analysis, for example, by using clustering algorithms or genome assembly programs. ID tags are used to identify sample origins when biological samples are mixed before sequencing.

Dataset (2): isotype assignment of immunoglobulin 454 sequence data. Immunoglobulins (Igs) play a central role in the human immune response. Immunoglobulin genes are created through a series of genomic recombinations that bring together a number of smaller genes to create the functional rearranged genes. An Ig protein consists of two heavy and two light chains and is broadly divided into constant and variable regions. An immunoglobulin's isotype is determined by the gene sequence which encodes the constant region. In human there are five isotypes; IgA, IgD, IgG, IgE and IgM.

In the data here evaluated, the isotypes IgE and IgG are of interest. The IgE isotype is a key component in type I allergic reactions [18]. The IgG isotype is mainly directed against invading pathogens and includes six polymorphic sub-types. The issue addressed here is to confidently classify sequence isotypes as either IgG or IgE. The portions of the constant regions captured with 454 sequencing reads are relatively short: 32 bases for the IgE and 76 bases for the six IgG isotypes. Assignment of isotype is required for downstream analyses of mutation spectras that explore the roles of the different Igs in immune responses. For example, it has been observed that non-allergic IgE sequences have significantly less mutations than allergic IgE and IgG's [19].

3.3 Materials and methods

3.3.1 Hardware and software

CUDA is a C/C++ extension created by NVIDIA. The development kit and the CUDA drivers are freely available from NVIDIA (www.nvidia.com). PaSWAS requires at least CUDA version 3.1 and GPU hardware version 1.2. The minimum requirements to run PaSWAS are a standard desktop computer or laptop with a recent, low-cost, consumer-grade NVIDIA-based graphics card. Development and testing were done on a single GTX285 fitted into a computer with an Intel Core 2 Quad CPU running at 2.40GHz with 4GB of memory. The development machine also holds a GTX295, with two cores each a fraction slower than a GTX285, combined almost twice as fast. Further testing was also undertaken on a high performance computer with two C1060s and two Intel Xeon CPU X5650s running at 2.67GHz, 96GB memory and 7 TB storage. The processing environment utilized Condor [20], to provide a single grid with 19 GPU cores and 240 CPU nodes available in a single grid. The grid included fifteen GTX285, one GT295 and two C1060 Tesla graphics cards.

3.3.2 Smith-Waterman algorithm

The Smith-Waterman (SW) algorithm [5] is used in sequence analysis to find local alignments between two sequences. It requires a dynamic programming approach. In its naive form it is of order $O(n^2)$ for both computational resources and memory. For memory, this can be reduced to O(n) by storing only two rows from the alignment matrix [5]: during calculation of the alignment score only the previous row and the current row of the matrix are stored, together with the location of the highest value. When the calculations are finished, this location together with the highest value, the final alignment score, is returned. This process is used for searching through large databases when only the best hits are relevant and the alignment profile is not necessary in downstream analyses.

For the exact local alignment an alignment matrix needs to be calculated (Figure 3.1A). This procedure starts in the upper-left corner and steps through the matrix left to right, top to bottom and ends bottom-right. The location of the maximum value in the matrix indicates the end of the local alignment and this alignment needs to be traced back from this cell to the start.

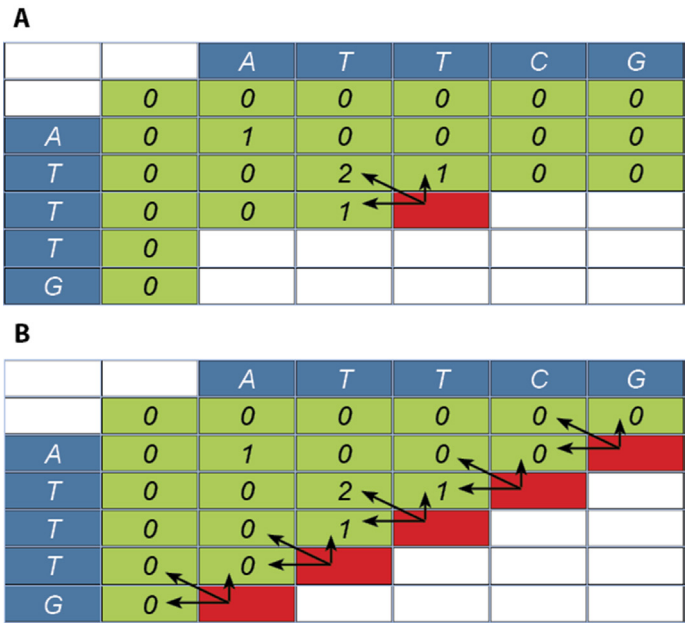

Figure 3.1. Smith-Waterman approach. A. Calculating a score for the alignment starts at the top-left and ends at the bottom-right of the matrix. The red cell is the next score that can be calculated with the arrows indicating the cells used. **B.** Parallel Smith-Waterman approach. With this algorithm the score of several cells is calculated in parallel (red cells). The arrows point to the cells used for the calculation of each new score.

3.3.3 Earlier parallel Smith-Waterman implementations

There are several parallel SW implementations available [12,14,15,17]. These implementations are primarily focused on searching sequence databases. They do not produce alignment profiles, which makes them fast and memory efficient but prevents visual inspection of the results. Other implementations [21,22] are capable of producing the alignment profiles, but show only one profile per alignment. These parallel SW implementations focus on two distinct properties of how the alignment matrix is filled (Figure 3.1B). There is a diagonal dependency of cells: cell at position (x,y) can only be processed when cells (x-1,y), (x,y-1) and (x-1,y-1) are known. In the first step only one thread is active, for cell (0,0). In each subsequent step an additional thread becomes active and halfway through the matrix the maximum number of threads are active: minimum(n,m). The number of active threads then decreases to a single thread for the last step at (n-1,m-1). This can be made more efficient by using idle threads to work on different sequence alignments [17]. In case of many sequence alignments, the processing units will be very active. For larger sequences this advantage disappears. In case the processor can calculate only a single alignment because of the length of the two sequences, there are no other cells to update and threads will remain idle. PaSWAS uses a similar approach in calculating the scores and tracebacks.

3.4 Results

3.4.1 Parallel Smith-Waterman Alignment Software (PaSWAS)

PaSWAS consists of three separate phases: (1) calculation of alignment scores, (2) determination of tracebacks and (3) production of profiles as output of the results. These steps are outlined here and the implementations are explained in more detail in the next sections. For the explanation of the application the following setup is used. On the horizontal axis there are X number of sequences, each of length N. These may be, for example reads from a sequencing platform. If a sequence is shorter than N, it is padded to length N with a special character. All sequences are placed in a single string x of length X*N. On the vertical axis the target sequences are placed. There are Y target sequences, each of length M. These sequences are padded when shorter than M. They are placed in a single string y of length Y*M.

In the first phase, the alignment matrix of each sequence alignment is calculated. The strings x and y are copied to the main (global) memory of the GPU. Each sequence alignment is calculated in parallel. Each alignment is updated over the diagonal of the matrix and starts at the top-left. At the start there will be X*Y threads active and at peak performance there are X*Y*minimum(N,M) threads active. During the entire phase the maximum value is tracked. This value is necessary to decide which tracebacks need to be calculated.

In the second phase, the traceback of each alignment is determined. Based on user-defined settings, such as the maximum value in the matrix required, the tracebacks are calculated in parallel. This implies the reverse order of the previous phase, starting at the bottom-right (Figure 3.1B). The profiles are stored in main memory using host page-locked memory. This keeps the data transfer between CPU and GPU to a minimum and there is no need to claim additional memory on the GPU.

The last phase is run on the CPU and consists of producing the alignment profiles. Because of the parallel nature of the algorithm, the profiles are presented in the output in random order. It is, therefore, not possible to rely on the order of the input when parsing the output. The output contains additional information about each profile, including the number of gaps, mismatches and the start and end of the alignments.

3.4.2 Phase 1: calculation of alignment matrix

The GPU contains two types of memory: global memory and shared memory. Global memory is the main memory and is usually several hundred megabytes up-to 12 gigabytes in size. Shared memory is distributed across the GPU and is located physically close to the processors. It is relative small in size, but also much faster to access than global memory. Global memory is used to store the scores, the directional matrix and the strings. Because this is relatively slow memory access is therefore minimized by using the much faster shared memory on the GPU. Although shared memory can only contain 16 kilobytes per block, each of these memory blocks is shared amongst threads in the same block.. For this reason, intermediate results during processing of the alignments and scores are stored in shared memory.

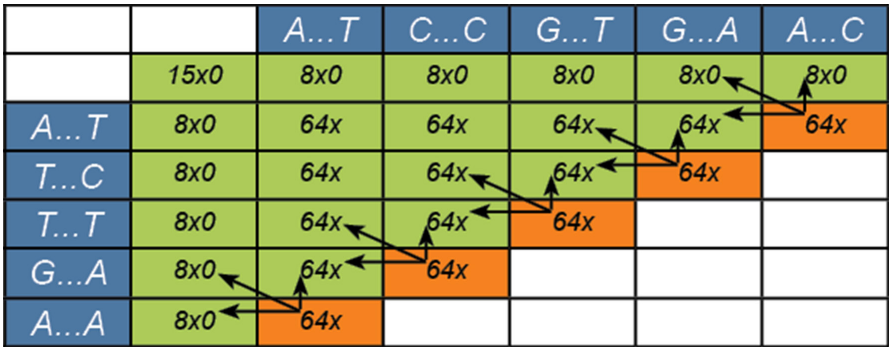

Figure 3.2. Subdivision of the alignment in PaSWAS. Each cell indicates a sequence alignment of 8 characters on the X-axis and Y-axis. The orange blocks are calculated in parallel in the same way as is depicted in Figure 3.1B. The arrows indicate the blocks required for the block being calculated.

Each sequence alignment is subdivided into smaller matrices of 8x8 cells (Figure 3.2). The occupancy calculator provided by NVIDIA (www.nvidia.com) indicates that an 8x8 block is the most efficient setting to make optimal use of the current hardware and tests confirm these settings (see Figure 3.6.S1). These 8x8 matrices map to thread blocks of 64 threads. The eight characters of the two sequences, the scores and the maximum value are stored using shared memory. This requires several data transfers from the host to global memory: the characters, the scores and maximum values from the surrounding blocks are retrieved from global memory. Without scores from a neighboring block, for example if it is the first block, scores are initialized to zero. This is all calculated in parallel.

Within each block, calculations start at the top-left and pass through the matrix via the diagonal to the bottom-right. To make use of idle threads in a block, the maximum value is determined during this pass as well. Upon completion, the resulting information is copied to global memory.

Similar to the cells within the matrix, each block depends on the three surrounding blocks. At the start, X*Y blocks of 64 threads are launched. The maximum number of thread blocks launched is (X*N*Y*M) / 64. For example, 100 reads of 400bp versus 100 sequences of 500bp will launch between 10,000 and 31,250,000 blocks at any given point in time, with each block having 64 threads. These numbers will vary between different types of graphics cards and depend on the amount of global memory available.

3.4.3 Phase 2: Determination of tracebacks

After the alignment is calculated, traceback is necessary to generate the alignment profiles based on the values in the matrix. If the maximum value within the alignment matrix complies with the requirements set by the user, it is marked as the start of the traceback. This includes a check of the value against a user-provided minimum value to give multiple profiles per alignment.

This phase is the opposite of the previous phase: the application starts at the bottom-right block and the bottom-right cell of this block. When a cell is the starting point of an alignment profile, the information about this starting point is copied to the memory of the computer (host). The x, y and score are copied, as well as the direction the score was coming from: the cell located at the left, up or upper-left. These starting points are calculated in parallel as well, which is taken into account when copying the data to the host. A procedure has been implemented to ensure that each starting point is stored without forcing the program into a sequential flow, which would otherwise slow down execution. On the host an array is allocated to store the starting points. The GPU has an integer index which points to the first available position. When a thread detects a starting point this index is increased by one using the atomic function of the hardware [23]. This function guarantees that this thread is the only process accessing this value. The host also has a matrix allocated to store the direction. This directional value is also copied directly to the host.

The thread now marks the location where the score comes from as negative in the matrix. The SW algorithm requires that all values are larger than, or equal to, zero. A negative sign is therefore appropriate to mark the traceback on the GPU without the need for additional memory space. Each thread mapped to a cell therefore checks not only for starting points, but also for negative values. If a negative value is found, the direction is copied to the host and the score at that location will be marked as negative. At the end of each block all negative values are copied to global memory for the surrounding blocks to continue the traceback until the block in the upper-left corner is reached.

Property	Example value	Explanation
Id X	GI3U2RK01APIT7	Identification label of the sequence
Id Y	IgG2_02	Isotype label of the sequence
Relative score	2.353	Alignment score / alignment length
Alignment length	68	Total length of the alignment profile, including gaps
Gaps	10	Number of gaps in the alignment profile
Matches	48	Number of matches in the alignment profile
X start	383	Location in X where the alignment starts
X end	449	Location in X where the alignment ends
Y start	1	Location in Y where the alignment starts
Y end	59	Location in Y where the alignment ends
Score	160	Score of the alignment profile

```
X: CCTCCAC-CAAGGGCCCA-TCGGTCTTCCCCCTGG-CGCC-CTGCTCCAGGAGCACCTCCGAGAGCAC
   ||||||| || |.|||||| ||.|||||||||.|.| | || |||||.||..|.||. ||  |. ||||
Y: CCTCCACACA-GAGCCCACTCCGTCTTCCCCTTCGAC-CCGCTGCTGCAAAAACAT-TC--AC-GCAC
```

Figure 3.3. Output of PaSWAS for a single alignment. The property column gives the name of the property available, followed by an example of a value for each property. The last row shows the alignment profile of X versus Y with '|' indicating a match, '-' a gap and '.' a mismatch.

3.4.4 Phase 3: Production of profiles

This phase is relatively short. The list produced in phase 2 contains the locations in the direction matrix where to start the traceback. This list is now processed. The application goes through this matrix using the directional information and counts the number of gaps, mismatches and matches and creates three strings: the sequence, the target with the gaps marked and the alignment profile. An example of the output is shown in Figure 3.3.

3.4.5 Test case 1: Tag recovery

In the first test case, PaSWAS is used for short adaptor and tag detection and subsequent cleaning of a set of sequence reads. The data set contains 401,824 reads with lengths between 50 and 600 nucleotides from a 454 GS Titanium platform (www.roche.com), comprising 28 different samples labeled by a sequence tag that defines the origin of each read. Each read has the following structure:

[5' ID tag][5' primer sequence][genomic sequence][3' primer sequence][3' ID tag]

The 5' ID tag is five nucleotides long. The neighboring 5' primer sequence is either 19 or 21 bases long and is used for sample identification. The 3' primer sequence and 3' ID tag have the same functionality as their 5' counterparts. In this study only the results of the 5' data are used. The 3' data is only used to support the 5' analysis and is not used for primary identification.

Table 3.1. ID tag recovery in a 454 data set.

Data	PaSWAS results
Number of reads processed	401,824
Number of primers used	2
Time (sec)	92
Recovery 5' ID tags	357,395 (88.9% of reads)

Results of ID tag recovery in a 454 data set obtained with PaSWAS.

Identification of the exact start site of the primer is necessary to retrieve the tag and the exact end of the primer is necessary to prevent contamination of the genomic sequence. During the sequencing process small numbers of errors can be introduced. A read may therefore contain gaps, nucleotide substitutions and nucleotide additions compared to the primers and tags used. Perfect sequence matching of every read is therefore unlikely. Adding or deleting parts of the genomic sequence makes detecting small changes such as SNPs in the genomic sequence more difficult.

PaSWAS was used to align the 5' primer sequences to the reads and the resulting alignments were used to get the tag immediately adjacent to this primer sequence.

Including the reverse complement of the primer sequence, this resulted in 1,607,296 sequence alignments. PaSWAS ran for 92 seconds, calculating 17,470 alignments per second (roughly 1.2 giga cell updates per second). In the 401,824 reads 357,395 5' ID tags were identified by PaSWAS (Table 3.1).

The performance of PaSWAS (Table 3.2) is based on a relative score. This score is defined as the alignment score divided by the alignment length. The match score is set to 5.0, so a relative score of 5.0 indicates a perfect match over the entire alignment. In this data set, 44.9% of reported

alignments contain the entire primer sequence with no mismatches and/or gaps. There are 53.5% hits with a relative score of 5.0. 8.6% of the hits have mismatches/gaps at the start or end of the primer and a full match over the local alignment (53.5-44.9). A substantial number of alignments have a less-than-perfect match, which shows that detecting only perfect matches over the entire length of the primer will result in substantial data loss.

Table 3.2. Performance of PaSWAS in ID tag recovery.

Description	Relative score (score / length)	Hits (%)
Full primer recovery	X= 5.0	44.9
Perfect alignments	X= 5.0	53.5
Some gaps / mismatches	$4.0 \leq X \leq 5.0$	35.3
Low quality alignments	$3.0 \leq X \leq 4.0$	10.5
Very low quality alignments	$X \leq 3.0$	0.6

44.9% Of the hits contains the full primer sequence. PaSWAS presents an accurate alignment tool able to retrieve degenerated tags. The relative score used is defined as the alignment score divided by the alignment length.

```
7   ACT--CAG--GCTGAGATGG  22   FSGIRH301EX46I
    | |  |||  |||||.||||
1   A-TGGCAGCAGCTGAAATGG  19   1-1

19  ATGGCAGCAGCTGAAATGG   37   FSGIRH301EX46I
    ||||||||||||||||||
1   ATGGCAGCAGCTGAAATGG   19   1-1
```

Figure 3.4. Example sequence alignment with multiple hits per alignment. In this figure the alignment of a primer sequence (1-1) to a 454 read (FSGIRH301EX46I) is shown. The top profile shows the location of degenerated primer. The bottom profile shows the best alignment, starting at position 19 in the read.

A possible source of errors is a primer that matches at more places, either because of sequencing errors or by chance. PaSWAS is the only GPGPU-based SW implementation now available which allows to investigate such cases. PaSWAS is able to produce multiple hits per sequence alignment and is therefore capable of detecting the correct location of the primer. Figure 3.4 shows the location of the primer in the top alignment profile and the best hit, starting at location 19 in the read, in the bottom alignment. In this case the second-best hit allows for proper tag detection in this read.

The results presented for this case show that the accurate alignment accomplished by PaSWAS has considerably added value for retrieving degenerated tags.

3.4.6 Test case 2: Isotype assignment of immunoglobulin genes

To show the flexibility of PaSWAS with different data types, we used an immunoglobulin data set. Given short (32-76 bases) sequences of immunoglobulin constant regions, the aim is to classify the correct isotype of the immunoglobulins for downstream analysis of the spectra of mutations.

The 454 data set consisted of 55,295 reads with candidate IgE and IgG sequences. Of these, the IgE isotype classification required confirmation. This confirmation was based on comparison with the known IgE and IgG sequences.

On a GTX285, it took PaSWAS 154 seconds to perform the 774,130 alignments (including reverse complement), representing a rate of 5,026 alignments per second or 72 mega cell updates per second.

Table 3.3. Classification of immunoglobulin sequences by PaSWAS.

Immunoglobulin classification	IgE	IgG	Unclassified	Total
Classified by PaSWAS	32,947	17,505	4,843	55,295

The table shows the number of sequences classified by PaSWAS as either IgE or IgG. A small subset of the dataset (11.4%) could not be classified as either IgE or IgG.

Table 3.4. Number of mutations found in classified immunoglobulin IgE and IgG isotypes.

	Identified by PaSWAS
IgE	
Total mutations	89,895
Total unique sequences	7,120
IgG	
Total mutations	115,335
Total unique sequences	6,109

For both the isotypes IgE and IgG the total number of mutations and number of unique sequences identified with PaSWAS is given.

PaSWAS identified 32,947 IgE sequences and 17,505 IgG sequences in the data set (Table 3.3). As a consequence, there are 4,843 sequences for which the isotype could not be confirmed. For the sequences classified, the number of unique sequences and the number of mutations compared to the sequences of the known isotypes (IgE or IgG) was determined with PaSWAS. In Table 3.4, the number of mutations and unique sequences identified are presented. For the IgE sequences, 89,895 mutations became available for analysis and for the IgG sequences 115,335 mutations. Figure 3.5 shows an example of the alignment profile generated by PaSWAS that is essential for such mutational analyses.

Both cases presented show the applicability of PaSWAS. It is able to handle real-life sized data sets fast enough while delivering the accuracy of a full SW.

```
1   CCTCCACACAGA-GCCCATCCGTCTTCCCCTTG 32
    |||||||||||| |||. ||||||||||||||
356 CCTCCACACAGACGCCG-TCCGTCTTCCCCTTG 387
```

Figure 3.5. Alignment generated by PaSWAS. The top entry is part of the sequence of the IgE constant region and the bottom entry is part of a 454 read. A dot indicates a mismatch in the alignment and a minus symbol indicates a gap. Mismatches are clearly distinguishable in the profile, with two gaps and one mismatch.

3.5 Discussion and conclusions

In this paper we present PaSWAS as novel parallel implementation of the SW algorithm on a GPU. There are several advantages of PaSWAS over other SW implementations. The first advantage is that PaSWAS produces complete alignment profiles with gaps, mismatches and matches and is not limited to one hit per sequence alignment. To our knowledge, the parallel SW implementations currently put forward in the literature do not allow inspection of the actual alignments and do not present multiple hits per alignment. PaSWAS can, if so desired, deliver a virtually unlimited number of local alignments for a single sequence comparison (Figure 3.4). This is essential for comparing short sequences (reads) to long sequences such as genomic regions.

PaSWAS is not limited to particular types or lengths of sequences. It can handle RNA and protein sequences just as easily as DNA sequences and allows the use of different scoring matrices. Moreover, PaSWAS is relatively fast.

Both test cases presented show that the innate accuracy of the SW algorithm as implemented in PaSWAS can give added value such as detailed alignment information necessary for SNP or degenerated tag detection . The example only shows the practical applicability of PaSWAS for this type of application. Other software such as Reaper [24] or Trimmomatic [25] may be as suitable or better than PaSWAS for tag detection. Proper benchmarking would require more and more detailed comparisons of different software. This is considered a challenge for the future.

The current output of sequencing technology combined with the speed of PaSWAS implies that thousands to millions of sequences can be recovered and/or identified for further analysis, resulting in efficient use of sequence data and effective analysis.

PaSWAS is accurate and fast, but for large genome-wide analysis, for example analyses of millions of sequence reads of the human genome, a cluster of GPUs is still required to make computations feasible in time. Each year the number of floating point operations per second performed by a GPU is increasing rapidly, so such genome-wide analyses will become feasible in the near future.

The added value of PaSWAS is particularly in producing alignment profiles. These profiles can be inspected visually or automatically for gaps and mismatches to allow, for example, SNP detection. To allow for multiple hits per sequence alignment and additional profile information, PaSWAS requires additional storage and significantly more calculations compared to searching through a database. It is therefore inappropriate to compare the speed of this algorithm with search-only GPU-applications. PaSWAS is slower due to the additional calculations and memory access. PaSWAS currently focuses on local alignments. Features to be included in the future are gap extension penalties, codon insertion/deletion scoring and a user-friendly framework so the application can be easily plugged into existing analysis pipelines. This is likely to result in different versions of the software, because we expect that some of these features will present a performance penalty. Such a penalty may not be attractive for all uses or users. Other features will be faster than calculating local alignments, because they will require significantly less administration and calculations.

A revised version of PaSWAS is under development (Chapter 4). The current implementation is based on the vender-specific CUDA platform. This limits the use of PaSWAS to NVIDIA-based graphics cards. These cards are widely available but to make PaSWAS run on other brands of GPUs, on different types of CPUs and on other many-core architectures, we are currently working on an OpenCL (https://www.khronos.org/opencl/) implementation of PaSWAS.

With such developments, the use of the SW algorithm on GPUs and CPUs will continue to present even more attractive approaches for the analyses of (next generation) sequence data.

3.6 Acknowledgements

We would like to thank Harold Verstegen (formerly KeyGene NV, now at KWS SAAT AG) for fruitful discussions and making the DNA tag data and other resources available, as well as Bruno Gaeta (School of Computer Science and Engineering), Alan Wilton († 2011) and Andrew Collins (School of Biotechnology and Biomolecular Sciences), all from the University of New South Wales, who also helped with data, discussions and resources. We would also like to thank the three anonymous reviewers for their comments, which helped us to improve the manuscript considerably.

3.7 References

1. Metzker ML. Sequencing technologies - the next generation. Nat. Rev. Genet. 2010;11:31–46.
2. Chan EY. Next-generation sequencing methods: impact of sequencing accuracy on SNP discovery. Methods Mol. Biol. 2009;578:95–111.
3. Naqvi AR, Islam MN, Choudhury NR, Haq QMR. The fascinating world of RNA interference. Int. J. Biol. Sci. 2009;5:97–117.
4. Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, et al. *De novo* assembly of human genomes with massively parallel short read sequencing. Genome Res. 2010;20:265–72.
5. Smith F. Comparison of biosequences. Adv. Appl. Math. 1981;2:482–9.
6. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J. Mol. Biol. 1990;215:403–10.
7. Kent WJ. BLAT--the BLAST-like alignment tool. Genome Res. 2002;12:656–64.
8. Emde AK, Grunert M, Weese D, Reinert K, Sperling SR. MicroRazerS: rapid alignment of small RNA reads. Bioinformatics. 2010;26:123–4.
9. Li R, Li Y, Kristiansen K, Wang J. SOAP: short oligonucleotide alignment program. Bioinformatics. 2008;24:713–4.
10. Li R, Li Y, Fang X, Yang H, Wang J, Kristiansen K, et al. SNP detection for massively parallel whole-genome resequencing. Genome Res. 2009;19:1124–32.
11. Farber RM. Topical perspective on massive threading and parallelism. J. Mol. Graph. Model. 2011;30:82–9.
12. Khajeh-Saeed A, Poole S, Blair Perot J. Acceleration of the Smith–Waterman algorithm using single and multiple graphics processors. J. Comput. Phys. 2010;229:4247–58.
13. Vouzis PD, Sahinidis N V. GPU-BLAST: using graphics processors to accelerate protein sequence alignment. Bioinformatics. 2011;27:182–8.
14. Liu Y, Maskell DL, Schmidt B. CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units. BMC Res. Notes. 2009;2:73.
15. Manavski SA, Valle G. CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment. BMC Bioinformatics. 2008;9:S10.
16. Schatz MC, Trapnell C, Delcher AL, Varshney A. High-throughput sequence alignment using Graphics Processing Units. BMC Bioinformatics. 2007;8:474.
17. Liu Y, Huang W, Johnson J, Vaidya S. GPU Accelerated Smith-Waterman. Int. Conf. Comput. Sci. 2006;3994:188–95.
18. Collins AM, Sewell WA, Edwards MR. Immunoglobulin gene rearrangement, repertoire diversity, and the allergic response. Pharmacol. Ther. 2003;100:157–70.
19. Dahlke I, Nott DJ, Ruhno J, Sewell WA, Collins AM. Antigen selection in the IgE response of allergic and nonallergic individuals. J. Allergy Clin. Immunol. 2006;117:1477–83.
20. Thain D, Tannenbaum T, Livny M. Distributed computing in practice: the Condor experience. Concurr. Comput. Pract. Exp. 2005;17:323–56.
21. Korpar M, Šikic M. SW#-GPU-enabled exact alignments on genome scale. Bioinformatics. 2013;29:2494–5.
22. de O. Sandes EF, de Melo ACMA. Retrieving Smith-Waterman alignments with optimizations for megabase biological sequences using GPU. IEEE Trans. Parallel Distrib. Syst. IEEE; 2013;24:1009–21.
23. NVIDIA. NVIDIA CUDA programming guide. NVIDIA Corporation; 2017.
24. Davis MPA, van Dongen S, Abreu-Goodger C, Bartonicek N, Enright AJ. Kraken: a set of tools for quality control and analysis of high-throughput sequence data. Methods. 2013;63:41–9.
25. Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics. 2014

3.8 Supporting information S1

3.8.1 Comparison of PaSWAS to BLAST-based approaches

To show the value of PaSWAS relative to ongoing procedures in laboratories, we here show the comparison of the results from PaSWAS with the results from routinely used BLAST-based approaches. The Blast-based analyses were used as standard in the labs we collaborated with. Such standard analysis should be considered as matter of fact; it may not be of have been the best possible solution for such analyses. Many more options for software are available. However, often biology-oriented laboratories stick to known procedures and do not go or dare to go into benchmarking or implementing new software. Although we show that PaSWAS outperforms these standards, the comparison here shown is not meant as benchmark of any available alternative software. As the comparison is not meant as or suitable as benchmark, default BLAST settings were used and the BLAST was not necessarily optimized for this particular case .The results only demonstrate that PaSWAS is suitable for such analyzes and PaSWAS out-performs the methods used routinely.

3.8.2 Tag recovery

PaSWAS is used for short adaptor and tag detection and subsequent cleaning of 454 sequence reads. The results are compared to the output and performance of a hardware-accelerated (FPGA) platform. This platform consists of dedicated hardware running a BLAST-like algorithm setup with default settings and is routinely used but not optimized for this particular case. Filtering of the data was performed after the BLAST analysis. Details of the data are given in the main text of the paper.

The number and identity of tags retrieved by PaSWAS were compared to the results of the hardware platform analysis. The results are given in Table 3.5 S1. PaSWAS ran for 92 seconds, calculating 17,470 alignments per second (roughly 1.2 giga cell updates per second). In the 401,824 reads 357,395 5' ID tags were identified by PaSWAS, compared to 287,995 by the BLAST-based hardware platform. These 287,995 tags are for 99.7% contained in the set of the ID tags identified by PaSWAS. Hence only 0.3% of the tags identified by the hardware platform are not identified by PaSWAS. Compared to the standard set-up, PaSWAS retrieves 19.5% more tags, involving 88,9% of the reads. As a consequence, considerably more useful information is extracted from the sequence data set.

3.8.3 Isotype assignment of immunoglobulin genes

Aim is to classify the correct isotype of the immunoglobulins for downstream analysis of the spectra of mutations based on the 32-76 bases short sequences of the Ig constant regions.

The result of the BLAST analysis used in the laboratory is compared to the results of a PaSWAS analysis. Details of the data are given in the main paper.

The 454 data set consisted of 55,295 reads with candidate IgE and IgG sequences. Of these, the IgE isotype classification required confirmation. This confirmation was based on comparison with the known IgE and IgG sequences. On a GTX285, it took 154 seconds to perform the 774,130 alignments (including reverse complement), representing a rate of 5,026 alignments per second or 72 mega cell updates per second. The BLAST analysis used for comparison was part of a larger grid application, that ran for several hours on a multicore grid. The pre- and post-processing was part of the analysis on the grid implementation used. The results are shown in Table 3.6 S2. PaSWAS identified 15.4% more IgE sequences and 5.3% more IgG sequences in the data set compared to the BLAST approach. As a consequence, there are 52.1% fewer sequences for which the isotype could not be confirmed. For the sequences classified, the number of unique sequences and the number of mutations compared to the sequences of the known isotypes (IgE or IgG) was also compared between PaSWAS and BLAST (Table 3.7 S3). For the IgE sequences, PaSWAS identified 8,906 (11.0%) more mutations for analysis and for the IgG sequences 3,001 (2.7%) more than the BLAST approach .Both laboratories were pleased with the way PaSWAS extracted more useful data from the data than the methods they were used to. For proper benchmarking, however, it will be interesting and necessary to compare the results of PaSWAS with the results of other dedicated software and validation in the lab.

Table 3.5. S1. ID tag recovery in a 454 data set.

	PaSWAS		Hardware platform
Number of reads		401,824	
Number of primers		2	
Time (sec)	92		120
Recovery 5' ID tags	357,395 (88.9% of reads)		287,995 (71.7% of reads)
Overlap		99.7%	
Relative performance		119.5%	

Comparison of the results of ID tag recovery in a 454 data set obtained with PaSWAS and the hardware platform.

Table 3.6. S2. Classification of immunoglobulin sequences by PaSWAS and BLAST.

Immunoglobulin classification	IgE	IgG	Unclassified	Total
PaSWAS	32,947	17,505	4,843	55,295
BLAST	28,549	16,628	10,118	55,295
% change	+15.4	+5.3	-52.1	

The percentage change indicates the difference in number of sequences classified by PaSWAS compared to the BLAST-based approach.

Table 3.7. S3. Number of mutations found in the classified immunoglobulin IgE and IgG isotype data.

IgE				
	PaSWAS	**BLAST**	**Difference**	**Difference (%)**
Total mutations	89895	80989	8906	11.0
Total unique sequences	7120	6387	733	11.5
IgG				
Total mutations	115335	112334	3001	2.7
Total unique sequences	6109	5917	192	2.0

For both the isotypes IgE and IgG the total number of mutations and number of unique sequences identified with either PaSWAS or BLAST is given.

3.8.4 Supporting information figure S1

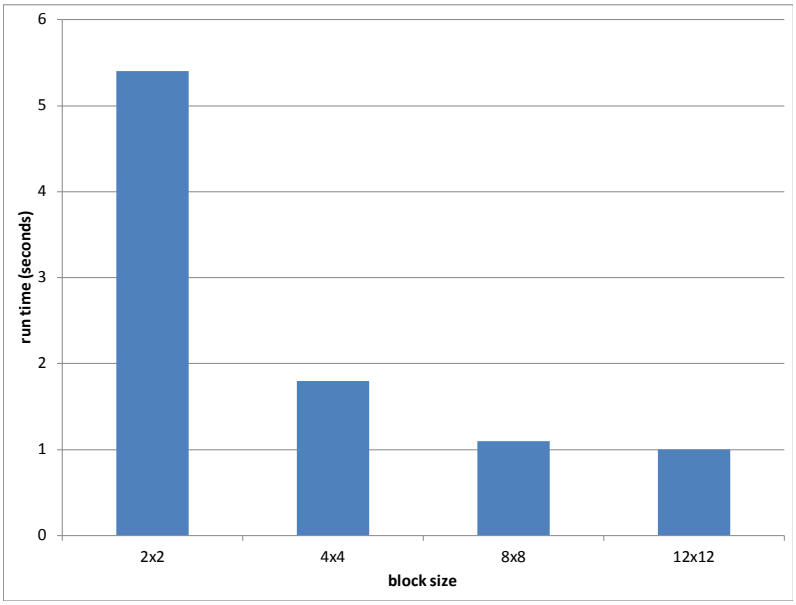

Figure 3.6. S1: Timing per block size. This plot shows the speed of the PaSWAS algorithm (y-axis) for different thread block sizes (x-axis).

4 pyPaSWAS: Python-based multi-core CPU and GPU sequence alignment

Published as S. Warris, N.R.N. Timal, M. Kempenaar, A.M. Poortinga, H. van de Geest, A.L. Varbanescu, J.P. Nap, "pyPaSWAS: Python-based multi-core CPU and GPU sequence alignment." *PLoS ONE* (2018) 13(1): e0190279. https://doi.org/10.1371/journal.pone.0190279

4.1 Abstract

Background

Our previously published CUDA-only application PaSWAS for Smith-Waterman (SW) sequence alignment of any type of sequence on NVIDIA-based GPUs is platform-specific and therefore adopted less than could be. The OpenCL language is supported more widely and allows use on a variety of hardware platforms. Moreover, there is a need to promote the adoption of parallel computing in bioinformatics by making its use and extension more simple through more and better application of high-level languages commonly used in bioinformatics, such as Python.

Results

The novel application pyPaSWAS presents the parallel SW sequence alignment code fully packed in Python. It is a generic SW implementation running on several hardware platforms with multi-core systems and/or GPUs that provides accurate sequence alignments that also can be inspected for alignment details. Additionally, pyPaSWAS support the affine gap penalty. Python libraries are used for automated system configuration, I/O and logging. This way, the Python environment will stimulate further extension and use of pyPaSWAS.

Conclusions

pyPaSWAS presents an easy Python-based environment for accurate and retrievable parallel SW sequence alignments on GPUs and multi-core systems. The strategy of integrating Python with high-performance parallel compute languages to create a developer- and user-friendly environment should be considered for other computationally intensive bioinformatics algorithms.

4.2 Background

A major challenge in applied bioinformatics is the adoption of advanced high-performance tools and algorithms by end-users with possibly low-to-moderate software engineering skills in the context of their biological research questions. Earlier, we presented the CUDA-only application PaSWAS (Chapter 3) that performs Smith-Waterman (SW) sequence alignment for any type of sequence on NVIDIA-based GPUs [1]. PaSWAS is relatively fast and combined the accuracy of SW alignment with the possibility to retrieve alignment information relevant for biologists, in contrast to most other parallel SW implementations. Yet, adoption of PaSWAS can be improved: it may be too complex to install and use. In addition, use of the application was limited to NVIDIA-based hardware. Also in other cases, the adoption of highly promising tools and approaches is slower than expected. For example, the *de novo* assembly tool CloudBrush [2] uses MapReduce on Hadoop [3,4], but has seen no biological applications yet. The three versions of the NVIDIA CUDA-based sequence alignment tool CUDASW++ [5–7] are cited often, but citations deal in

the larger majority with novel software implementations. The latest version CUDASW++ 3 [7], for example, has been cited 116 times (as of July 2017) but none of these citations deal with a direct biological question. The lack of adoption of promising new developments in algorithms and hardware may indicate that we as developers underestimated the complexity of setting up and running such a new application, especially when it is limited to a certain platform.

Another important limiting factor in the use of PaSWAS is the absence of the affine gap penalty. This scoring method produces biologically more relevant alignments than using only a gap open penalty [8]. It is therefore an important feature missing from the Smith-Waterman implementation in PaSWAS.

To improve the accessibility and use of PaSWAS, we have developed an entirely new software package, pyPaSWAS, based on OpenCL and CUDA integrated with Python. Python is a platform-independent programming language, with many libraries appropriate for bioinformatics, such as BioPython [9] and SciPy [10]. The open compute language OpenCL [11] is the current standard for clusters and/or multi-core CPU/GPU's to speed-up analyses up to several orders of magnitude compared to single core CPU versions. OpenCL is similar to CUDA, but is supported by a growing number of manufacturers, including Intel, NVIDIA, Apple and IBM. By supporting both CUDA and OpenCL, pyPaSWAS runs on many platforms, including CPUs, GPUs other than NVIDIA-based GPUs and so-called accelerator cards. We integrated the PaSWAS CUDA (Chapter 3) and OpenCL codebases with Python through pyCUDA [12] and pyOpenCL [12]. The original PaSWAS code was extended to add support for the affine gap penalty scoring method [8]. The result is a versatile Python-based user-friendly application for SW sequence alignment on a variety of multi-core systems. We propose this strategy as showcase for the integration of new software based on these compute languages with common programming tools such as Python to promote the adoption of advanced tools and applications in applied bioinformatics.

4.3 Implementation

The new software package pyPaSWAS is implemented in Python (2.7 and up) and is run from the command line. It uses the libraries pyOpenCL [12] and pyCUDA [12] for device handling, memory allocation and kernel invocations to run the core PaSWAS Smith-Waterman code on the parallel device. pyPaSWAS depends on OpenCL 1.2+ [11] or Cuda 2.0+ [13], numpy[14] and biopython [9]. All other processing, such as Input / Output handling, logging and exception handling, are done in standard Python. The SeqIO class from bioPython [9] is used for file input. Its reference manual [15] lists all formats supported, including multi-fasta, genbank and fastq. Input file formats not supported by bioPython can be implemented by extending the Core.Reader class. Output can be formatted in a custom format by extending the Core.DefaultFormatter class. The Core.SAMFormatter class generates SAM output and can also be used as template

for other custom output. The SAM descriptors (Table 4.1) are particularly useful for further processing output data. File-based configurations allow for storing settings and consistent reruns of the application. The user can supply appropriate scoring values for alignment, for example substitution matrices, to adjust the analyses to the desired specifications. The Core.Score module can be adjusted to support any 255 by 255 scoring matrix. The accompanying wiki [16] provides a complete description of the command line arguments as well as examples of how to run pyPaSWAS.

The structure of CPU hardware differs from GPU hardware and running OpenCL code designed for GPUs is not optimal [17]. Therefore, two OpenCL versions based on the CUDA-based implementation in PaSWAS were developed, one for GPUs and one for CPUs [18]. The latter makes better use of CPU hardware for faster sequence alignments. The two OpenCL implementations differ from the previous CUDA implementation only in the use of specific OpenCL calls; no changes have been made to the underlying algorithms.

The OpenCL implementation runs on multi-core hardware supporting OpenCL 1.2, such as Intel/AMD CPUs and accelerator cards (GPUs and Xeon Phi). With the CUDA implementation, pyPaSWAS runs on all NVIDIA GPUs with compute capability 1.2 and above, which includes support for all recent NVIDIA GPUs, including laptop versions, Teslas and the GTX-based cores. By default, pyPaSWAS runs on the CPU using the CPU-optimized OpenCL code. To use other parallel devices than the CPU, the user changes the configuration or selects the appropriate device through command line options.

pyPaSWAS opens the platform selected, sets the appropriate memory usage and other parameters relevant for the parallel device automatically, based on settings and data to be analyzed. pyPaSWAS allows for fine grained control over the use of the parallel device, such as memory usage and number of compute cores to be used. CPU hardware allows for limiting the number of cores used by an application. This enables using the computer for other tasks and is necessary when pyPaSWAS runs in a cluster environment. This fine-grained control level presents a major improvement over the earlier PaSWAS (Chapter 3) [1] in addition to the integration with Python. All options are listed on the wiki-page [16] and are accessible through the command line (‘-h’).

As its predecessor (Chapter 3) [1], pyPaSWAS documents all alignment details and allows for filtering of the resulting alignments. Parameters for filtering are listed in Table 4.1. Parameters can be set through the configuration file or through command line options. This gives the ability to select which hits are relevant and will be sent to the output file. The scoring value and all related values, such as query coverage, are present in the output and can also be used to filter the results further afterwards (Table 4.1).

4.3.1 Affine gap penalty

For biologically more relevant alignments, the affine gap penalty method [8] scores the opening of a gap differently than for extending a gap. The original PaSWAS code only supported the gap penalty scoring method, which means that each gap has the same score, no matter how many gaps are in front of it. The affine gap penalty implementation requires a scoring matrix *M*, to keep track of the match scores and scoring matrices *I* and *J* to keep track of the scores for gaps in the target (*I*) and query (*J*) sequences. The PaSWAS implementation of the direction matrix has been extended to record which of the three matrices resulted in the highest score. The downside of using an affine gap method is that it requires creating two additional matrices (*I*, *J*) of the same size as the already existing scorings matrix (*M*). This means that a 100x100 sequence alignment using the affine gap requires not 10,000 scoring values, but 30,000 scorings values. Next to an increase in memory usage, additional calculations compared to the original SW implementation are needed, making the affine gap method slower (see S3). The affine gap penalty method is required in all cases, for example when the gaps originated from technical (NGS) issues and do not have any biological meaning. In such cases, the PaSWAS code is used to perform a SW-alignment without a gap extension penalty. The user controls the use of the affine gap penalties by setting a value other than zero for the gap extension penalty (the '-g' option).

Table 4.1. Options in PyPaSWAS for selecting and filtering the alignments.

Filter name*	Value range**	Default	SAM descriptor	Description
lower_ limit_score	0.0 < x <= 1.0	1.0		Allows for more hits per alignment. All hits with a score within this fraction of the maximum score found are reported. Used during the backtracing procedure for reducing the number of alignments to be processed.
minimum_ score	0 < x	30	AS:i:	Minimum score of an alignment. Used during the backtracing procedure for reducing the number alignments to be processed.
filter_factor	0.0 < x <= 1.0	0.2	AS:i:	For each alignment the theoretical maximum score is calculated: length of the shortest sequence times the maximum score for a match (eg. the score for a perfect alignment). Only alignments with a score above filter_factor times this theoretical maximum score are returned.
query_ coverage	0.0 <= x <= 1.0	0.2	QC:f:	Minimum fraction of the query covered in the alignment
query_ identity	0.0 <= x <= 1.0	0.2	QI:f:	Minimum fraction of matches relative to the query
relative_ score	0.0 < x <= score match	2.0	RS:f:	Minimum score relative to the shortest sequence. A full match will give a relative score of the match score, for DNA/RNA sequences the default is 5.0
base_score	0.0 < x <= score match	2.0	BS:f:	Score of the alignment divided by the length of the alignment.

*Filter name: all parameters available for filtering; ** value range: the boundaries for the settings of the corresponding parameter.

4.4 Results and discussion

The performance of pyPaSWAS is expressed as the time required for the number of SW alignments processed. Six different configurations were tested for performance (Table 4.2), with variations in hardware (Intel or NVIDIA), parallel device (CPU or GPU), code usage (optimized for CPU or GPU), number of cores used and the language involved (OpenCL or CUDA).

Table 4.2. Configurations for testing the performance of pyPaSWAS.

Configu-ration	Hardware	Parallel device	Code optimized for	Nr. of cores	Language	Time for 2720 alignments (s)	GCUPS*	Speedup compared to F
A	Intel i7	CPU	CPU	1	OpenCL	119.2	0.70	0.21
B				8		106.4	0.82	0.21
C			GPU	1		812.6	0.10	0.03
D				8		192.3	0.44	0.12
E	NVIDIA GTX 1070	GPU	GPU	1920	OpenCL	57.8	1.48	0.36
F**					CUDA	17.6	4.64	1.00

*GCUPS: giga cell updates per second. **Configuration (F) is equivalent to the earlier PaSWAS [1], and is therefore used as reference here. The last two columns give the amount of time spent on the largest set of alignments in the performance analysis and the speedup compared to the configuration (F).

In all cases, pyPaSWAS was run on a standard desktop (Intel i7 -2600K) running Ubuntu 16.02 and holding an NVIDIA GeForce GTX 1070 GPU. Timing of alignments was done by determining the run time of the application between first and last API calls to the Python libraries (either pyOpenCL or pyCUDA), so overhead such as file I/O is not taken into account. The full report is in Supporting Information S1. Performance analysis with the same data set on a standard laptop is in Supporting Information S2.

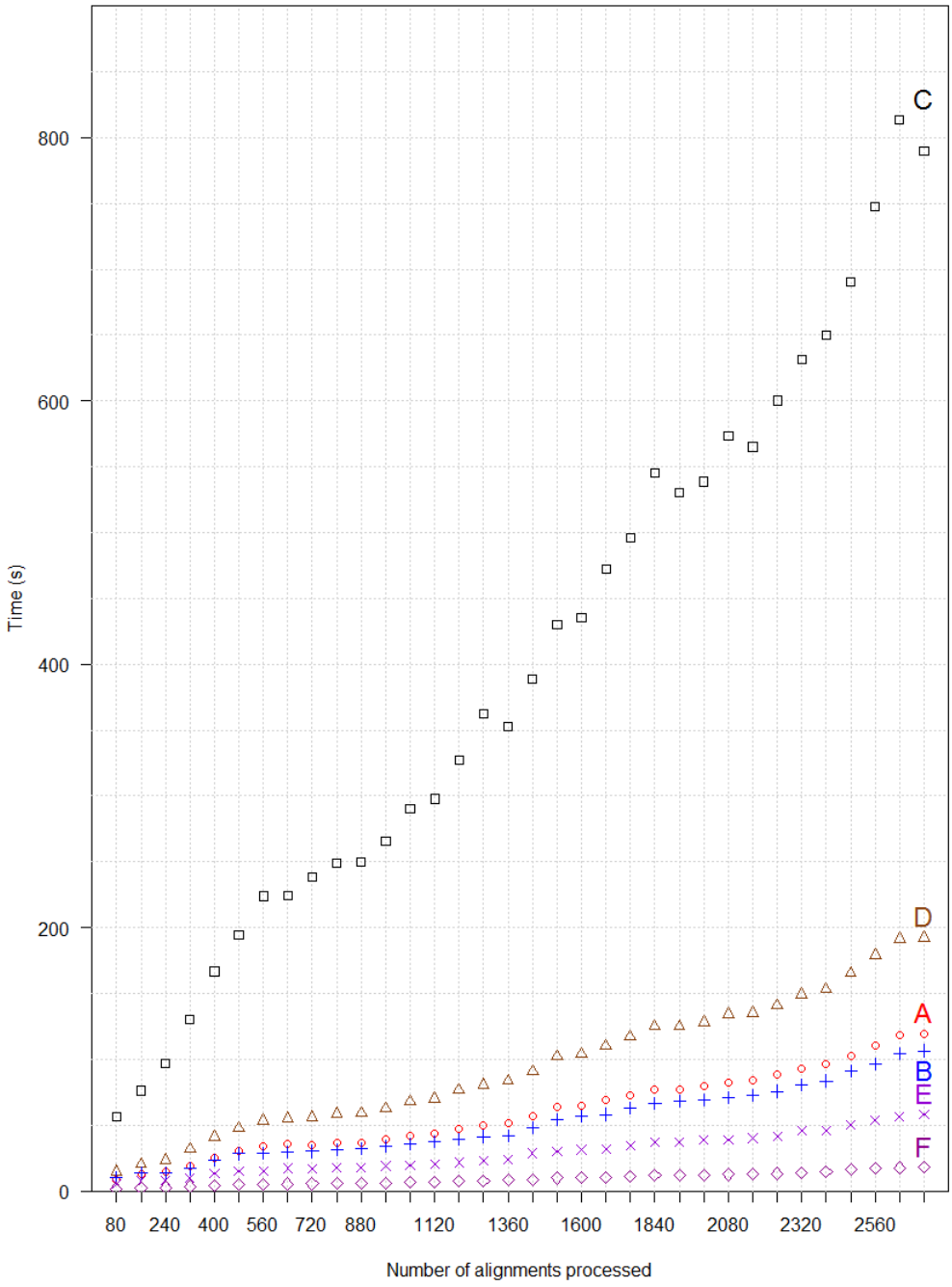

Figure 4.1. Performance of six different configurations for pyPaSWAS in Smith Waterman (SW) alignments. The time required (Y-axis) for processing an incremental number of alignments (X-axis) is plotted. For details of the different configurations A-F see Table 4.2.

As test set for the performance analysis of pyPaSWAS on the different hardware configurations, the Ankyrin repeat protein set from the domestic dog (*Canis lupus familiaris*; CanFAM 3.1, GCA_000002285.2), consisting of 348 proteins was used. For the performance analysis, the eight proteins not labeled 'PRED' were selected and aligned to an increasing number of proteins from the total data set. The time required to calculate the increasing number of SW alignments by the six configurations is shown in Figure 4.1. The time for performing the maximum of 2720 sequence alignments is also given in Table 4.2. As these protein sequences differ in length, it is common to indicate the speed of the SW computations in giga cell updates per second (GCUPS) to create an performance indicator independent of sequence length. The alignment output itself and the biological context were not considered. In this example data set the CUDA implementation running on the GPU (F) is the fastest configuration and is 2.8 times faster than the OpenCL version optimized for the GPU (E). The data also shows that the fastest configuration (F) is 33.3 times faster than the for GPU optimized OpenCL version on single CPU core, showing the advantages of parallel processing of SW alignments on a GPU. The for CPUs optimized OpenCL version (B) is 1.8 times faster than the for GPUs optimized version (D) on the same CPU. This shows that creating an OpenCL version of an application optimized for a particular hardware platform can speed up the application further. The performance tests using only a single core demonstrate the ability of pyPaSWAS to scale-down the number of cores used for the sequence alignments. The CUDA version (configuration F) is faster than the OpenCL version on a GPU (configuration E) , showing the added value of having a CUDA version in this case. There are several other reasons for having CUDA support in pyPaSWAS. In general, CUDA is faster than OpenCL [17]. Also, on some systems we tested, notably Apple Macs, OpenCL is not fully supported on NVIDIA GPUs, so CUDA is the only option available. Furthermore, several NVIDIA GPU products support only 32 bits memory allocation for OpenCL, which limits the amount of usable memory to 2 GB, but allow 64 bits memory for CUDA.

Analyses of the impact of the affine gap penalty on overall performance when the using the same data sets show that, on a desktop PC, all configurations are slower: from 1.14 times to 2.0 times slower (Supporting Information S3). Combined with the fact that memory requirement is also three times higher, it is therefore opportune to make sure that the affine gap is relevant for the task at hand.

A major advantage of PaSWAS for biological analyses is that it documents all alignment details necessary for further analysis, in marked contrast to other parallel SW implementations that focus on computational speed of the best alignment (Chapter 3) [1]. When for example compared to CUDASW++ version 3.0 getting the alignment profile comes with a performance penalty of about 25x (119.0 GCUPS [7] compared to 4.64 GCUPS) on similar hardware. The novel implementation pyPaSWAS here presented is more versatile for biological analysis then the original PaSWAS

code-base: not only full alignment details are stored and available for inspection, it also allows for gap extension penalties in scoring the alignment. In addition, the output can now also be formatted as a SAM file. Also, pyPaSWAS has more command line options and the output contains more relevant information, such as query coverage and query identity scores. The Python codebase enables bioinformatics researchers to add other output formats, store the alignments directly in a database or connect the application with workflow systems such as Galaxy [19]. In addition, the source repository holds configuration files to build Docker containers, including one Docker container with CUDA and OpenCL support, to allow for easy installation of pyPaSWAS and the required drivers and libraries.

As data volumes continue to grow and analyses tend to become more complex in every branch of bioinformatics, the added value of advanced high-performance IT solutions such as multicore CPUs and GPUs is transforming into a need for such solutions. Multicore CPUs for Blast [20] and BWA [21], cluster computing for Interproscan [22] and cloud infrastructure for a wide range of biomedical/bioinformatics applications are available [23]. High performance technology used in mathematics [24,25] and audio/video processing [26] rely on GPUs and OpenCL. Wider acceptance of OpenCL -based GPU applications in bioinformatics is likely to be promoted by packaging the C++ code for parallelization in a much more common used language such as Python as demonstrated here. The pyPaSWAS integration of Python with OpenCL should promote further use of advanced algorithms in bioinformatics. Given this successful showcase for the integration of OpenCL with new or existing software in Python, it could be considered to port bioinformatics algorithms that make use of advanced high performance technology to Python, R [27], Matlab [28] or Java [29] in a way similar to pyPaSWAS. This will promote use, maintenance and development of high-performance implementations of bioinformatics applications further. Such an approach could benefit for example algorithms for genome wide association studies [30], eQTL analyses [31] or phylogenetics [32].

4.5 Conclusions

pyPaSWAS is the implementation in Python of a general-purpose SW alignment supporting both the basic gap penalty method as well as the affine gap penalty method. The application runs fast on many multi-core systems, including GPUs and Xeon Phis, while still offering the desired flexibility to inspect any given alignment and all its parameters. The Python-based application will increase the use and utility of the parallel SW approach of PaSWAS. The smooth integration of Python with the much more complex languages OpenCL and CUDA for parallel execution of the SW algorithm makes pyPaSWAS easier to develop and maintain than its predecessor. The relative ease of Python, as well as the much larger community of programmers in Python, is likely to promote adoption and use, as well as facilitate addition of novel features to pyPaSWAS.

4.6 Acknowledgements

We thank Tim te Beek (former Netherlands Bioinformatics Centre) and Shruti Srivastava (Wageningen University & Research) for support during the development process, as well as Piet Plomp (Hanze University of Applied Sciences) for developing and maintaining the IT infrastructure. This work was partly carried out on the Dutch national e-infrastructure with the support of SURF Cooperative. Dick de Ridder (Wageningen University & Research) was helpful in suggesting the performance experiments and discussing the results.

4.7 References

1. Warris S, Yalcin F, Jackson KJL, Nap JP. Flexible, Fast and accurate sequence alignment profiling on GPGPU with PaSWAS. PLoS One. 2015;10:e0122524.
2. Chang Y-J, Chen C-C, Ho J-M, Chen C-L. *De novo* assembly of high-throughput sequencing data with cloud computing and new Operations on string graphs. 2012 IEEE Fifth Int. Conf. Cloud Comput. IEEE; 2012. p. 155–61.
3. Hadoop - Apache Software Foundation project home page [Internet]. Available from: http://hadoop.apache.org/
4. Taylor RC, Baker M, Sansom C, Stein L, Schatz M, Langmead B, et al. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. BMC Bioinformatics. 2010;11 Suppl 1:S1.
5. Liu Y, Maskell DL, Schmidt B. CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units. BMC Res. Notes. 2009;2:73.
6. Liu Y, Schmidt B, Maskell DL. CUDASW++2.0: enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions. BMC Res. Notes. BioMed Central; 2010;3:93.
7. Liu Y, Wirawan A, Schmidt B. CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions. BMC Bioinformatics. 2013;14:117.
8. Gotoh O. An improved algorithm for matching biological sequences. J. Mol. Biol. 1982;162:705–8.
9. Cock PJA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. 2009;25:1422–3.
10. Jones E, Oliphant T, Peterson P. SciPy: Open source scientific tools for Python [Internet]. Available from: http://www.scipy.org
11. Munshi A, others. The opencl specification. Khronos OpenCL Work. Gr. p. 11-15; 2009;
12. Klöckner A, Pinto N, Lee Y, Catanzaro B, Ivanov P, Fasih A. PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. Parallel Comput. 2012;38:157–74.
13. NVIDIA. CUDA Download [Internet]. http://developer.nvidia.com/cuda-downloads.
14. NumPy [Internet]. http://numpy.scipy.org/.
15. bioPython [Internet]. http://biopython.org/wiki/Biopython.
16. Warris S. pyPaSWAS Wiki [Internet]. Available from: https://github.com/swarris/pyPaSWAS/wiki
17. Fang J, Varbanescu AL, Sips H. A Comprehensive performance comparison of CUDA and OpenCL. 2011 Int. Conf. Parallel Process. IEEE; 2011. p. 216–25.
18. Timal NRN. Accelerating protein sequence alignment with different parallel hardware platforms (MSc Thesis). Delft University of Technology; 2015.
19. Goecks J, Nekrutenko A, Taylor J. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. 2010;11:R86.
20. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J. Mol. Biol. 1990;215:403–10.
21. Li H, Durbin R. Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics. 2010;26:589–95.
22. Zdobnov EM, Apweiler R. InterProScan - an integration platform for the signature-recognition methods in InterPro. Bioinformatics. 2001;17:847–8.

23. Luo J, Wu M, Gopukumar D, Zhao Y. Big Data application in biomedical research and health care: a literature review. Biomed. Inform. Insights. 2016;8:1.

24. Demidov D, Ahnert K, Rupp K, Gottschling P. Programming CUDA and OpenCL: a case study using modern C++ libraries. SIAM J. Sci. Comput. Society for Industrial and Applied Mathematics; 2013;35:C453–72.

25. OpenCL Libraries and toolkits [Internet]. Available from: http://www.iwocl.org/resources/opencl-libraries-and-toolkits/

26. Kola G, Kosar T, Livny M. A fully automated fault-tolerant system for distributed video processing and off-site replication. Proc. 14th Int. Work. Netw. Oper. Syst. Support Digit. Audio Video. Kinsale, Ireland; 2004.

27. Urbanek S. R OpenCL [Internet]. Available from: https://cran.r-project.org/web/packages/OpenCL/index.html

28. MathWorks. MathWorks GPU Computing [Internet]. Available from: http://nl.mathworks.com/discovery/matlab-gpu.html

29. Jocl.org. JOCL [Internet]. [cited 2016 Sep 1]. Available from: http://www.jocl.org/

30. Standish KA, Carland TM, Lockwood GK, Pfeiffer W, Tatineni M, Huang CC, et al. Group-based variant calling leveraging next-generation supercomputing for large-scale whole-genome sequencing studies. BMC Bioinformatics. 2015;16:304.

31. Jansen RC, Nap JP. Genetical genomics: the added value from segregation. Trends Genet. 2001;17:388–91.

32. Stivala AD, Stuckey PJ, Wirth AI. Fast and accurate protein substructure searching with simulated annealing and GPUs. BMC Bioinformatics. 2010;11:446.

5 Correcting palindromes in long reads after whole-genome amplification

Published as S. Warris, E. Schijlen, H. van de Geest, R. Vegesna, T. Hesselink, B. te Lintel Hekkert, G. Sanchez Perez, P. Medvedev, K. D. Makova, D. de Ridder, "Correcting palindromes in long reads after whole-genome amplification", *BMC Genomics* (2018) 19:798, https://doi.org/10.1186/s12864-018-5164-1

5.1 Abstract

Background

Next-generation sequencing requires sufficient DNA to be available. If limited, whole-genome amplification is applied to generate additional amounts of DNA. Such amplification often results in many chimeric DNA fragments, in particular artificial palindromic sequences, which limit the usefulness of long sequencing reads.

Results

Here, we present Pacasus, a tool for correcting such errors. Two datasets show that it markedly improves read mapping and *de novo* assembly, yielding results similar to these that would be obtained with non-amplified DNA.

Conclusions

With Pacasus long-read technologies become available for sequencing targets with very small amounts of DNA, such as single cells or even single chromosomes.

5.2 Background

Modern sequencers require sufficient material to work with: the Illumina and Pacific Bioscience (PacBio) platforms prescribe at least three micrograms, but recommend at least five [1] micrograms. Long-read sequencing technologies such as those offered by PacBio and Oxford Nanopore Technology (ONT) additionally require high molecular weight (HMW) DNA as a starting material, i.e. material in which individual DNA stretches are long. In many biological settings, obtaining sufficient amounts of DNA of the required quality and length is problematic, such as in studies on single cells [2,3] or single selected chromosomes [4]. To overcome this limitation DNA is amplified, starting from as little as picograms, in a process called whole-genome amplification (WGA) [5].

A major issue with the WGA process is that it introduces specific chimeric fragments [6,7] consisting of one or more inverted repeats (Figure 5.1), so-called palindromes. This effect is partially alleviated by de-branching, however, chimeric fragments still remain [8]. In Illumina paired-end (PE) sequencing these fragments are then sheared into small sub-fragments before library preparation, which reduces the effect on subsequent analyses of the palindromic fragments as they will occur in only few reads. In other approaches to sequencing, however, the full fragments are used. For Illumina mate-pair (MP) sequencing, long palindromic fragments will result in pairs with incorrect directions and unpredictable insert sizes. As a result, short read MP libraries based on WGA are problematic for read mapping and *de novo* assembly.

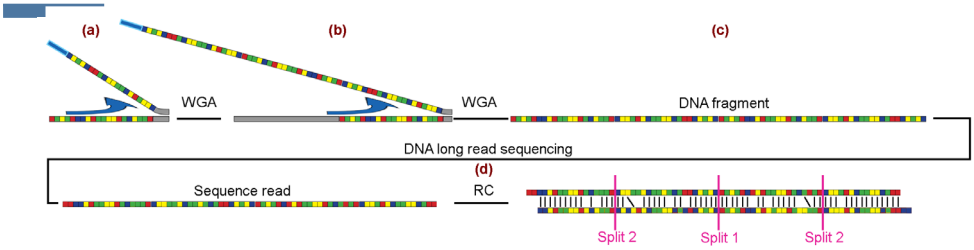

Figure 5.1. The introduction of palindromes by whole-genome amplification (WGA) and correction of these sequences with Pacasus. The colored squares in this figure indicate the four different nucleotides. In whole-genome amplification a DNA-polymerase binds to the DNA and starts making a copy of that strand (left-side of the image). Palindromes are introduced when during WGA the DNA-polymerase continues with elongation (indicated by the arrow) along an already created WGA product **(a)**, generating a palindrome. In this example this incorrect elongation occurs several times **(b)**, resulting in a DNA fragment containing four copies of the original fragment **(c)**, which is sequenced. Pacasus detects the palindrome sequence by aligning the read's reverse complement to itself **(d)** and splits the read in two smaller reads at the center of the alignment (split 1). This process is repeated and splits the two resulting reads again (split 2), yielding four separate, 'clean', reads. The full set of reads, corrected and left intact, is then used in, for example, read mapping or *de novo* assembly.

Tools specifically aiming to detect and correct chimeric reads have been proposed (e.g. uchime [9]) and work well for paired-end and single-end short-read technologies. For long reads however, the palindromic nature of the sequence hinders read mapping and renders *de novo* assembly highly problematic. Due to the high base-calling error rate of the long read technologies (11-38%) [10,11], finding and correcting these palindromic constructs in long reads cannot be done by exact string matching. Algorithms for improving long-read quality in general are available: Proovread [12], PBcR [13] and ECTools [14] use either Illumina HiSeq reads or assembled contigs based on HiSeq data to increase the quality of base calling. While Proovread can also detect chimeric fragments, it specifically aims at detecting PCR artifacts joining fragments originating from different regions in the genome. This is done by mapping HiSeq reads, assumed to be available, which do not have these chimeras. The location of the chimera in the long read is then detected by finding discrepancies in the short-read mappings. This approach is unfit for solving the chimeras occurring due to WGA: the HiSeq reads are based on the same fragments as the long reads and will therefore contain the same nucleotide sequence information. As a consequence of this lack of suitable methods to correct these chimeras, the use of WGA with long-read technologies was usually not advised [6], which precludes the application of long read technology to answer essential biological questions at the single-chromosome or single-cell level.

Here, we introduce a new method, *Pacasus*, for correcting palindromic, long, error-rich reads without the loss of nucleotide information and with only very limited impact on repeats and palindrome sequences of biological origin. The method relies on a Smith-Waterman alignment

implementation called pyPaSWAS [15,16], which supports fast processing on multicore CPUs, GPUs and Xeon Phis to detect palindromes and iteratively corrects them by splitting up reads. We demonstrate its performance on PacBio sequencing data of *Arabidopsis thaliana* as well as on flow-sorted gorilla Y chromosome data, by using the multiple displacement amplification kit REPLI-g for the amplification process.

The gorilla Y chromosome was selected because primate Y chromosomes are relatively short and contain many repeats, rendering them difficult to sequence and assemble. Even in one of the most complete assemblies, that of *Homo sapiens*, more than half of the sequence of Y is still unknown [17]. To obtain a higher read coverage of the gorilla Y chromosome, a recent paper [4] used flow-sorting and WGA. PacBio long reads, Illumina HiSeq PE and MP-libraries, transcriptome data and PCR sequences were used by the authors as well (Bioproject PRJNA293447). The RecoverY tool [4,18] presented in same paper was designed to identify short reads originating from the Y chromosome. Based on these data, the authors created a hybrid (PacBio + HiSeq) assembly, here labeled 'GorY'. The authors also used HGAP [19] and MHAP [20] to create PacBio-only assemblies, but these resulting assemblies were of suboptimal quality. In this manuscript, we used the PacBio data after WGA generated by Tomaszkiewicz and colleagues [4] to show the benefits of correcting palindrome sequences in this data set with an improved quality of the PacBio-only *de novo* genome assembly.

5.3 Results

5.3.1 Pacasus corrects many palindromic sequences found in WGA data

To demonstrate the added value of Pacasus in the analysis of PacBio reads generated from WGA DNA samples, we applied it to several data sets of *Arabidopsis thaliana* and a data set of the gorilla Y chromosome [4]. Table 5.1 lists the original number of reads, the number of reads that were found to have chimeras and the number of clean reads after correcting the palindromes. In the Arabidopsis samples, 40-50% of reads contained at least one palindrome, with some reads containing up to 15 (Figure 5.5 S1). This demonstrates the extent to which palindromes pose a problem in PacBio WGA data and illustrates that Pacasus effectively detects and corrects these.

Table 5.1 shows that Pacasus decreases the average read length of the *A. thaliana* set by about 48%, i.e. preserving much of the long-range information. In the gorilla read set, 11.8% of the reads contain palindromes, less than in the *A. thaliana* sets. The average length of the gorilla reads before processing with Pacasus is 5468b, i.e. 61.2% of the average length in the total Ath-WGA data set (8934b); after correcting the palindromes this is increased to 90.3%: 4234b compared to 4689b. Pacasus finds palindromes in only in 0.5% of the reads in the non-amplified control data set, Ath-Ctrl. These reads will be a mixture of false-positives and missed/missing SMRTBell

adaptors, which also cause palindromic sequences. The low number of palindromes found in the control set by Pacasus means that there is no need to perform subsequent analyses on 'Ath-Ctrl-Clean': the resulting *de novo* assembly for example will not be different from the one based on the original 'Ath-Ctrl' set.

The GC contents of the read sets were compared to that of the *A. thaliana* reference genome and no biases were observed for both the amplification process and the palindrome detection by Pacasus (Figure 5.2).

Table 5.1. Effect of correcting palindromes on number of reads and average read length.

Sample	Before cleaning		Reads with detectable palindromes		After correcting	
	Number of reads	Average length (b)	Number of reads	Number of reads (%)	Number of reads	Average length (b)
Ath-WGA1	462,138	9,326	221,001	47.8	869,826	4,660
Ath-WGA2	447,364	8,544	195,263	43.6	769,027	4,721
Ath-Ctrl	940,162	5,680	4,714	0.5	938,196	5,681
GorY-WGA	3,596,236	5,468	426,188	11.8	4,546,488	4,234

Effect of correcting palindromes on the number reads and average lengths of these reads. Note: the Ath-Ctrl shows a small increase in average read length after correction and a lower number of reads. This is because Pacasus removes very short reads from the output.

5.3.2 Correcting palindromes improves read mapping

Using the BLASR default settings and an additional filter of at least 80% nucleotide identity between read and reference, both the raw and clean read sets map well (Table 5.2). Palindromic reads map partially, leaving a (potentially large) proportion of the reads unmapped. This becomes clear when only mappings are considered where 80% and 95% of the complete read can be aligned: mapping efficiency for the raw read set drops from 99% to 44% and finally to 34%. For the clean reads, the mapping rates are 99%, 81% and 66% respectively, higher than for the noWGA read set (95%, 72% and 57%). Average coverages show similar effects. These mappings statistics indicate that the clean reads map more accurately and with higher read coverage than the raw reads do. The complete mapping reports are presented in Supplementary materials 2-4.

To verify the palindromic nature of the reads, the locations of the clean reads were also investigated. If the raw reads indeed contain palindromic sequences, the parts of the clean reads should map to the same region in the genome (in contrast to chimeric reads, where the parts originate from different regions in the genome). To verify this, the longest distance between the mapping locations of each part of the corrected reads was calculated and related to the length of the original raw read. 96.5% of these mapping distances are within the read length of the original read, showing that most of the clean reads map to the same region in the genome and that the original raw reads indeed contain palindromes, not other types of chimeras.

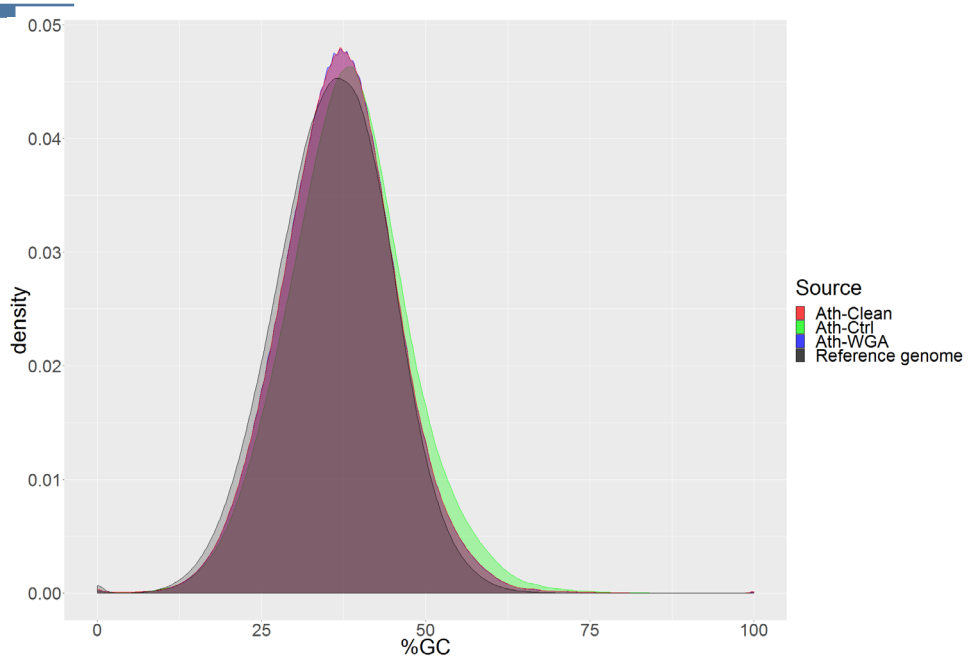

Figure 5.2. %GC density plot of Ath-Ctrl (green), Ath-WGA (blue), Ath-Clean (red) and the A. thaliana reference genome (black). The curves for Ath-WGA and Ath-Clean overlap completely. None of the three read sets show biases towards a certain GC-content when compared to the reference genome.

5.3.3 Assembly quality of corrected WGA reads approaches that of non-amplified data

To assess whether correcting palindromes also benefits assembly, we investigated two realistic scenarios using the *A. thaliana* data (Ath-WGA, Ath-Clean, Ath-Ctrl): PacBio-only assembly using Canu and a hybrid assembly, combining PacBio and Illumina HiSeq data, using DBG2OLC/Sparse. On the control data set Ath-Ctrl, this results in assemblies with overall good assembly statistics, with DBG2OLC yielding the best results (Table 5.3). Repeating the process with the original WGA data gives far worse results; the DBG2OLC assembly has, for example, an N50 value about 26-fold smaller than the N50 value of the control data and the assembly covers only about half (49.7%) of the reference genome.

Correcting the palindromic reads improves the hybrid assembly: although the N50 is lower than that of the Ath-Ctrl assembly, the assembly length and genome coverage are higher.

The Ath-Clean PacBio-only assembly is even better than the assembly based on the Ath-Ctrl data, with a higher N50 and genome coverage (Table 5.3). This is also reflected by the contig length distribution (Supplementary materials 5). Apparently, the removal of conflicting information outweighs the loss of long-range information.

The hybrid assembly and the PacBio-only assembly based on Ath-Clean are longer than the TAIR10 reference genome (119.7Mb), being 123.9Mb and 131.0Mb respectively. The full genome is thought to be approximately 135Mb [21], so this additional sequence information could be new genomic data. No further testing has been done to verify this.

Table 5.2. Read mapping statistics

	Reads mapped (%)			Average coverage			Average read length		
Alignment filter	**-**	**80%**	**95%**	**-**	**80%**	**95%**	**-**	**80%**	**95%**
Ath-WGA	99	44	34	40.5	17.4	13.4	8,987	5,568	5,397
Ath-Clean	99	81	66	55.1	48.1	41.2	4,690	4,532	4,697
Ath-Ctrl	95	72	57	34.2	29.2	24.8	5,799	5,583	5,852

Statistics of read mappings with BLASR to the TAIR10 reference genome, calculated without filtering for a minimum read alignment length ('-') and after filtering for reads aligned with at least 80% or 95% nucleotide identity.

Table 5.3. Statistics of the *de novo* assemblies

	PacBio-only (Canu)			Hybrid (DBG2OLC/Sparse)		
Read set	**Ath-Ctrl (C1)**	**Ath-WGA (C2)**	**Ath-Clean (C3)**	**Ath-Ctrl (D1)**	**Ath-WGA (D2)**	**Ath-Clean (D3)**
No. contigs	852	2,128	1,015	476	4,818	1,753
Length (Mbp)	115.6	116.8	123.9	110.9	108.9	131.0
Longest contig (Kbp)	1,181	655	3,402	5,667	246	2,239
GC (%)	36.0	36.2	36.12	35.97	36.57	36.21
N50 (Kbp)	293	73	302	823	32	278
L50	117	479	109	33	951	113
Covered (%)	86.6	91.2	97.3	85.1	49.7	96.3
Dupl. ratio	1.09	1.07	1.06	1.08	1.34	1.13

Statistics on the PacBio-only and hybrid assemblies of the various datasets. Note that the TAIR10 reference genome is 119.7 Mbp, with the full genome thought to be approximately 135 Mbp [21].

5.3.4 De novo assembly based solely on long reads of flow-sorted chromosomes is now possible

After correcting the palindromes in the original gorilla PacBio reads (see section 5.3.1) we were able to create two PacBio-only assemblies: GorY-WGA based on the raw data set and GorY-Clean, based on the clean reads. The GorY-WGA assembly was added to the comparison to stay in line with the *Arabidopsis thaliana* analyses described in the previous section and also to verify that the increase in quality is not only due to a better performing software application. Figure 5.3 shows the length distributions of both the contigs and the scaffolds in the previously published GorY hybrid assembly and the contigs in the new GorY-Clean / Gory-WGA PacBio-only assemblies. The GorY scaffolds were created by using long-range information to connect the contigs [4]. The scaffolds contain no additional information, except sequence contiguity. Gaps

between contigs in the scaffolds are filled with Ns. The top-10 longest contigs of GorY-Clean are as long as the top-10 longest GorY scaffolds (Figure 5.3), showing that the new contigs already contain the same contiguity except that the gaps are filled with sequence information. The scaffolded GorY assembly seems larger than the GorY-Clean one (Table 5.4, Supplementary materials 6). However, this is misleading as it contains 2.4 Mbp of Ns; the actual nucleotide content of the GorY assembly is 1.3 Mbp less than that of GorY-Clean. This is corroborated by further assembly statistics (Table 5.4). In terms of structure, the GorY-Clean assembly resembles the human Y chromosome assembly more than the original assembly (Supplementary materials 7). The GorY-WGA assembly is also of higher quality compared to the GorY contigs, but not as good as the GorY-Clean assembly. We attribute the quality increase of GorY-WGA compared to the GorY assembly to the use of Canu [22]; the improvement of GorY-Clean over GorY-WGA is most likely due to correcting the palindromic reads with Pacasus.

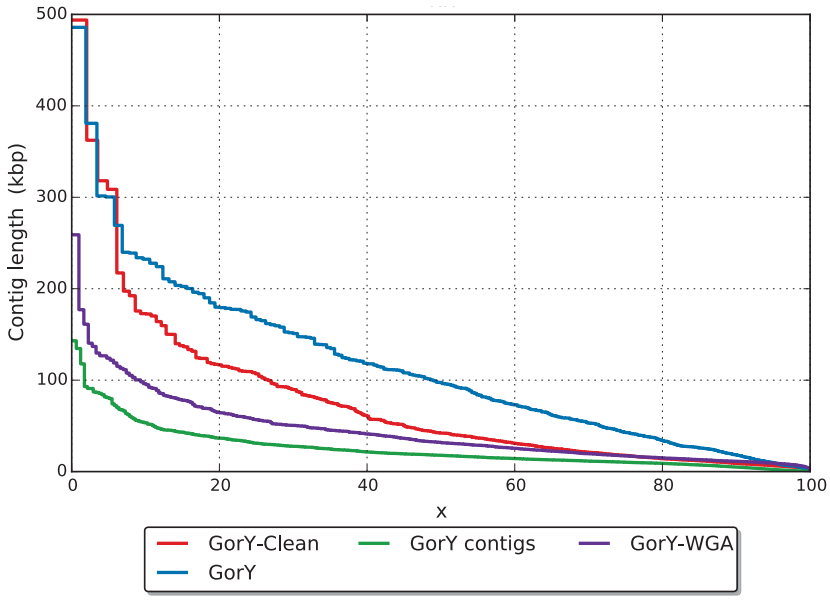

Figure 5.3. Contig length distributions. Contig length (y-axis) distribution of the published gorilla Y chromosome (GorY), the contigs underlying this assembly (GorY contigs), the *de novo* assembly based on raw PacBio data set (GorY-WGA) and of the *de novo* assembly of the cleaned reads (GorY-Clean). The x-axis shows the fraction of the assembly (e.g. the N20, N50, etcetera).

The accuracy of the newly constructed GorY-Clean contigs becomes more apparent when looking at the read mapping statistics (Table 5.5). To calculate these, only contigs are used as these contain sequence information: reads will not map to Ns in the scaffolds. The gorilla Illumina HiSeq reads map better to the human reference genome (HumY) than to the original GorY assembly (in terms of genome coverage) and overall mapping accuracy is highest for both newly created assemblies.

The GorY-Clean assembly is better covered by the read data than the GorY-WGA assembly is, regardless of whether corrected or non-corrected reads are used for evaluation. These results indicate that, apparently, currently available assemblers (in our case Canu) are better at handling chimeric reads than previous software and that the newly created assemblies (GorY-WGA and GorY-Clean) are more accurate than GorY.

The average coverage when using the raw reads increased from 73.15x to 83.21x for the GorY-WGA and GorY-Clean respectively and with the corrected reads from 97.08x to 109.67x. These results show that the *de novo* GorY-Clean assembly fits the read data best and, as seen with the *Arabidopsis thaliana* data, mapping accuracy increases after correcting the palindromic reads.

Table 5.4. Human and gorilla Y-chromosome assembly statistics

	Assembly size (Mbp)	Ns (Mbp)	Non-Ns (Mbp)	No. sequences	N50 (Kbp)	Longest seq. (Kbp)
HumY	57.2	30.4	26.8	1		57,227
GorY, contigs	23.0	0	23.0	3,001	18	143
GorY, scaffolds	25.4	2.4	23.0	697	98	486
GorY-WGA, contigs	26.5	0	26.5	1,128	32	256
GorY-Clean, contigs	24.3	0	24.3	1,062	42	494

Assembly statistics for the published human and gorilla Y chromosome assemblies and the new assemblies.

Table 5.5. Read mapping statistics on the human and different GorY assemblies.

		HiSeq			GorY-WGA			GorY-Clean		
Assembly	Length (Mbp)	Genome coverage		Read cov.	Genome coverage		Read cov.	Genome coverage		Read cov.
		(Mbp)	(%)		(Mbp)	(%)		(Mbp)	(%)	
HumY	26.8	22.3	83	1897	18.2	68	58.87	18.5	69	74.84
GorY	23.0	18.3	80	1169	21.1	92	71.33	21.1	92	99.15
GorY-WGA	26.5	24.9	94	1353	26.5	100	73.15	26.5	100	97.08
GorY-Clean	24.3	22.4	92	1586	24.3	100	83.21	24.3	100	109.67

Mapping of HiSeq, PacBio WGA and PacBio cleaned reads on the human Y chromosome (HumY), the gorilla Y chromosome (GorY) and the newly created gorilla Y assemblies (GorY-WGA, GorY-Clean). The read coverage is the average number of reads that a nucleotide has aligned to it.

5.3.5 Resolving artificial duplications provides a higher coverage of genes on the Gorilla Y chromosome

The gorilla Y chromosome contains 12 single-copy X-degenerate genes [23]. To evaluate completeness of these genes in the assemblies, their corresponding transcript sequences were mapped to the GorY and GorY-Clean assemblies using the mRNA aligner GMAP [24]. The resulting alignments were subsequently used to identify the contigs/scaffolds that harbor these genes. For these 12 genes, the transcript coverage was on average higher in GorY-Clean contigs (84.9%) than in GorY scaffolds (74.9%), while sequence identity was similar (Supplementary materials 8-9). Additionally, the complete (exons and introns) sequences of the orthologous genes in the human genome (GRCh38) were aligned to the contigs/scaffolds harboring these genes in the GorY and GorY-Clean assemblies. Visual inspection of the dotplots (Supplementary materials 10-15) identified fewer inversions and duplications in the GorY-Clean contigs than in the GorY scaffolds (Supplementary materials 16). In the alignment of the contigs from the GorY-Clean to the GorY scaffolds containing the same genes (Supplementary materials 17-22), no inverted duplications were detected in the GorY-Clean sequences. In contrast, numerous inverted duplications were visible in GorY sequences (Supplementary materials 23). Thus, many inverted duplications were resolved in the assembly generated from the sequencing reads corrected by Pacasus, suggesting that such duplications indeed are an artefact of WGA.

5.3.6 Effects on repeats and palindromic sequences of biological origin

DNA sequences are known to contain many different types and families of repeat sequences [25], including palindrome sequences [26]. These repeat sequences are present in the long reads after sequencing and can be detected by Pacasus as false positive palindromic sequences. When all repeats of biological origin are split, the subsequent *de novo* assembly will contain only collapsed regions effectively removing the repeats from the assembly. However, we speculate that most if not all of the true repetitive sequence are contained in sufficiently long reads to cover the repeats and therefore will not be considered by Pacasus for correction. In real data sets it is not known a priori which reads contain the true repetitive sequences, but evaluation is possible after performing the *de novo* assemblies: for both the *A. thaliana* and the gorilla Y chromosome the repeat content is known.

Table 5.6 shows the by RepeatMasker [27] detected repeat content of the *A. thaliana* assembly, including the reference assembly TAIR10. The repeat content of the assembly based on Ath-Clean (C3) is close to that of the reference and is higher than found in the assemblies based on Ath-Ctrl (C1) and Ath-WGA (C2), which is in line with the overall genome assembly statistics (Table 5.3).

To study the effect of Pacasus on actual biological palindromes present on the Y chromosome, 1-kb non-overlapping windows in GorY and GorY-Clean assemblies were identified which have high identity (>80%) to well-defined human chromosome Y palindromes P1-P8 (arm lengths 8.7-1450 kb) [28] and X-degenerate gene (XDG) regions. Since the two arms of each palindrome have high identity, the windows representing the palindrome arms should have twice the read depth compared to the windows overlapping with the single-copy XDG regions on the Y chromosome. In order to obtain the read depth of these regions, Illumina-based flow-sorted gorilla Y paired end reads were aligned to GorY and GorY-clean assembly, and the read depth for the windows specific to palindrome regions and XDG regions were extracted. Figure 5.4 shows that each palindrome is represented separately and Pacasus decreases the read depth for several palindromes, e.g., P3, P4, and P7. Nevertheless, the median read depth for each palindrome (except for P8) is still higher than XDG for both GorY and GorY-clean. This indicates that the biological palindromes are preserved by Pacasus.

Table 5.6. Repeat content of the different assemblies.

Assembly	Overall repeat content (% of assembly)
Canu Ath-Ctrl (C1)	15.73
Canu Ath-WGA (C2)	15.49
Canu Ath-Clean (C3)	16.76
TAIR10	16.88

Repeat content found by RepeatMasker in the different *A. thaliana* assemblies.

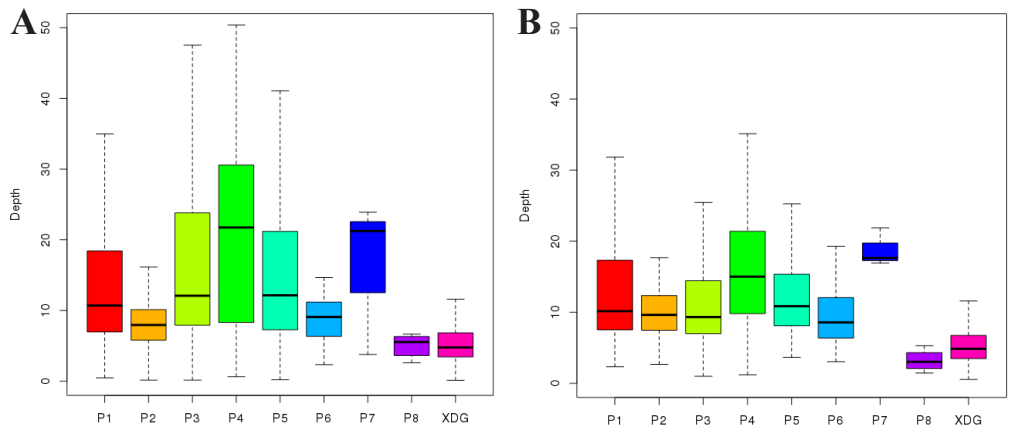

Figure 5.4. Illumina read depth of known palindromes. Illumina read depth of the known palindrome sequences P1-P8 and the X-degenerated gene (XDG) region in the GorY assembly (A) and GorY-Clean (B). Overal read depth is decrease in GorY-Clean, however in both assemblies the median read depths for P1-P7 are twice of that of XDG.

5.4 Discussion

After processing of the long reads, Pacasus has limited effect on the number of nucleotides in the read set and decreases the average read length by less than 50%. A downside of this process is that inverted repeats present in the genome will be treated as chimera, so that the repeat will be split into its separate elements, if the read does not span the full repeat. However, as shown in this manuscript, not all long reads suffer from chimeras. In most cases there will be sufficient reads long enough to cover the inverted repeat and palindrome sequences that are not split by Pacasus as shown in the assessment of the *A. thaliana* assemblies. It should be noted that Y chromosomes naturally possess non-artificial palindromes [28] and our analyses show that these palindromic sequences are also present in the *de novo* assembly after processing the reads by Pacasus.

Flow-sorted chromosomal DNA is usually contaminated with DNA from other chromosomes. Also with the gorilla sample, the original estimate is that approximately a third of the reads originate from the Y chromosome [4]. This is supported by our results, with 1,742,887 PacBio reads out of 4,546,488 reads mapping to the GorY-Clean assembly (38%). Consequently, some of the assembled contigs are not part of the gorilla Y chromosome but are from other chromosomes. Further analyses need to be performed to verify which contigs indeed originate from the Y chromosome. One suggestion is to look at read coverage: high coverage could point to Y chromosome sequences (see Suppl. Materials Figure 4d). The next step to further improve the quality of the assembly could be to scaffold the contigs using the RNA-Seq data from the previous study [4] with for example SSPACE [29] and polish the final assembly with the HiSeq paired-end data using Pilon [30].

Tissue-specific analysis at the genome level is becoming more important in, for example, studying cancer genomes, but for genome assembly approaches they are currently limited to short-read sequencing [31] and hence result in more fragmented assemblies than is possible with long reads. For polyploid plant species is possible to select pollen and extract DNA from these cells, effectively decreasing the ploidy by half and therefor also decreasing the complexity of subsequent assembly process. And recent research shows that CRISPR-Cas9 introduces unwanted changes in the genome best detectable by long-read sequencing [32]. By combining WGA and Pacasus on these types of tissues it is now possible to isolate low amounts of DNA and to produce a high-quality genome to find these alterations in the genome.

A possible application not discussed in this paper is the detection of a SMRTbell adapter that is missed by the PacBio software pipeline, producing a raw read with the same structure as created with WGA. These incorrect reads, although perhaps present in low numbers, can have an impact on quality of the *de novo* assembly. When a non-WGA PacBio dataset with high genome coverage produces a fragmented assembly, it is worthwhile to run Pacasus on this dataset to correct the

palindrome sequences created due to the missed SMRTbell adapter

The detection of the palindrome sequences requires a full Smith-Waterman alignment due to the high error rate of the long-read technologies, which takes a considerable amount of compute power. Using high performance software and several GPUs we were able to process one SMRTcell per day, roughly keeping pace with sequencing speed of the PacBio RSII. The throughput of the PacBio Sequel is higher, hence processing these SMRTcells will require more time or compute resources, but we believe the results presented in this manuscript warrant the investment.

To find the location in the read at which it needs to be split, the backtrace part of the Smith-Waterman alignment algorithm needs to be performed [15]. In the current implementation of the PaSWAS module used for SW, the memory requirements are quadratic in the length of the read. For reads above 100kb this memory requirement may limit the use of Pacasus. Currently the PacBio platforms generate reads below this length, but we expect the Oxford Nanopore platforms to go beyond this limit for at least some the reads in the near future. We will continue to work on Pacasus to decrease the memory requirements of the software to ensure that future output of sequencing platforms can be handled properly. The presented settings for the SW alignment are based on the error model of the RSII and our in-house experience with PacBio sequencing read qualities. For application on PacBio Sequel and Oxford Nanopore data, their respective error models may warrant minor changes to these settings.

5.5 Conclusions

Whole-genome amplification is required for sequencing when a biological sample contains insufficient DNA for direct use in library preparation, but the process creates chimeric fragments. We have developed a new method, Pacasus, to correct long, error-rich reads containing such chimeras, based on high-speed Smith-Waterman alignment. We demonstrated the performance of Pacasus in terms of read mapping accuracy and assembly quality, showing that the loss in read length is clearly offset by the removal of incorrect contiguity information. On the *Arabidopsis* data, the hybrid assembly improves markedly in quality; and on the gorilla data, a PacBio-only assembly on clean reads is even of higher quality than a hybrid assembly including the original reads. The differences between the GorY-Clean and GorY-WGA assemblies are, however, not as large as in the *A. thaliana* case. The underlying reason for this is the much lower number of detectable palindromes in the gorilla read set: 11.8% of the reads contain palindromes, compared to 45.8% of the reads in the *A. thaliana* set. Correcting the relatively low number of reads containing palindromes in the gorilla data set already gave an improvement in assembly statistics, which indicates that the impact of these incorrect reads on the assembly quality is high. We expect that longer reads contain more palindromes, as indicated by the differences in average lengths before and after correcting in both examples.

In summary, Pacasus now allows to analyze PacBio data obtained from low amounts of DNA, making it possible to apply the power of long read technology to, for example, the study of single cells (e.g. in cancer research) and the study of single chromosomes (also in polyploid organisms).

5.6 Material and methods

5.6.1 The Pacasus algorithm

To detect chimeras created during WGA, raw PacBio reads are aligned to their reverse-complement sequence with Smith-Waterman (Figure 5.1) using pyPaSWAS [16,33]. The parameters used for alignment are: gap score, -3; match score, 3; mismatch score, -4. For filtering, the parameters are: filter factor, 0.01; query coverage, 0.01; query identity, 0.01; relative score, 0.01; and base score, 1.0. When no overlap is found in the alignment, the read is left intact and stored in the output file; otherwise the read is split at the center of alignment (see Figure 5.1(d)). Both resulting fragments are again processed as if they were original reads, to allow the detection of nested palindrome sequences, until no overlap is detected anymore. If a fragment becomes shorter than a minimum length (default 50 bp) it is discarded. Note that the nucleotide information in the reads is neither removed nor changed. Pacasus is implemented in Python 2.7 and, besides pyPaSWAS (version >= 2.0), depends on Biopython [34] (version >= 1.67), numpy (http://numpy.org) (version >= 1.8.0) and scipy [35] (version >= 0.12.0).

5.6.2 Data for *Arabidopsis* evaluation

DNA was isolated from two *Arabidopsis thaliana* plants, labeled "Ath-WGA1" and "Ath-WGA2", and amplified using the REPLI-g Mini Kit (QIAGEN Benelux BV, Venlo, The Netherlands). The Arabidopsis thaliana are in-house samples based on low-input plant materials (for more details on DNA isolation, library preparation and sequencing see [36]). Both samples were sequenced on both an Illumina HiSeq2000 sequencer and a PacBio RSII sequencer. A third DNA sample was used to generate PacBio RSII data without WGA ("Ath-Ctrl"). Table 5.7 shows the number of reads generated by each platform and for each sample. To evaluate mapping performance, PacBio reads were mapped to the TAIR10 *Arabidopsis thaliana Columbia* reference genome (http://www.arabidopsis.org) using BLASR version 1.3.1.124201 [37], and alignments with identity < 80% were filtered out by a Python script. Mapping reports were generated in CLCBio version 8.0.2 (http://www.clcbio.com).

Table 5.7. Datasets used for the performance analysis of Pacasus.

Species	Sample	WGA	Illumina HiSeq2000		PacBio RSII	
			reads	length	reads	avg. length
Arabidopsis thaliana	Ath-WGA1	yes	31,233,196	100	462,138	9,326
	Ath-WGA2	yes	43,810,780	100	447,364	8,544
	Ath-Ctrl	no			940,162	5,680
Gorilla gorilla	GorY-WGA	yes	279,601,852	150	3,596,236	5,468

PacBio-only *de novo* assemblies of the *A. thaliana* genome were created using Canu version 1.3 [22]. Hybrid assemblies, combining the HiSeq2000 and RSII data, were created with DBG2OLC (released in 2016) [38]. DBG2OLC requires as input a HiSeq-only assembly; which we created using Sparse (released in 2015) [39], as recommended on the website by the authors of DGB2OLC, based on the HiSeq data from the WGA samples in all cases. The assembly was finalized with the PacBio reads.

We combined Ath-WGA1 and Ath-WGA2 into a single set, Ath-WGA, and created assemblies combining the HiSeq contigs with the raw Ath-WGA reads, with corrected (or 'clean') Ath-WGA reads (Ath-Clean) and with Ath-Ctrl reads. To evaluate quality, assemblies were compared to the reference genome using QUAST version 4.3 [40].

5.6.3 Data for the gorilla Y chromosome evaluation

PacBio RSII data of a flow-sorted and amplified gorilla Y chromosome, GorY-WGA (Table 5.7), was downloaded from the NCBI Short Read Archive (SRA SRX1161235). The previously published assembly of the gorilla Y chromosome and the publicly available data from the flow-sorted, whole-genome amplified and de-branched gorilla Y chromosome, GorY [4], were downloaded as well (GCA_001484535.2). Canu version 1.3 was used for the assembly of the PacBio reads. For comparison, the human chromosome Y assembly (NC_000024.10), HumY, was downloaded. QUAST [40] was used for assembly comparison and statistics. PacBio reads were mapped to HumY, GorY and the new assemblies using BLASR (>80% identity and >80% read coverage); Illumina HiSeq 2500 PE reads (SRA SRR2176191) were mapped using CLCBio version 8.0.2. Statistics for all mapping results were calculated in CLCBio. For calculating the contig length distribution of the GorY assembly, scaffolds were broken up and N's were removed. The gorilla X-degenerate gene transcripts were retrieved from a previous study [23]. GMAP version 2017-03-17 [24] was used to align the transcripts to the assemblies.

5.6.4 Repeats and palindrome detection

RepeatMasker [27] was configured with rmblastn (2.2.27+) [41,42] and RepBase (20140131) [43] for masking the *A. thaliana* assemblies and TAIR10 reference genome.

Following Tomaszkiewicz et al. [4] , the gorilla Y contigs were broken into 1-kb windows and each

window was aligned to human reference hg38 using lastz [44], with settings --scores=human_primate.q, --seed=match12, --markend. RepeatMasker was also run on gorilla Y contigs to mask repeats and later for each window the total number of masked sites 'N' within a window were calculated. The windows which overlap with human Y chromosome palindromes and XDG genes were identified and filtered to make sure that they have at least 80% match with the human reference and less than 20% of N's throughout the window.

BWA mem [45] was used to align the flow-sorted gorilla Y paired-end reads (SRX1160374) to the GorY-clean and GorY assemblies (unmasked). The bedtools [46] coverage function was used to calculate the read depth and coverage of each window. If the windows had > 80% coverage they were used to create boxplots within their respective palindromes. The boxplots were generated using R boxplot command with outline=TRUE parameter set.

The Human_primate.q file used for primate to primate alignments in lastz is as below:

```
gap_open_penalty   =   500 # O
gap_extend_penalty =    30 # E
hsp_threshold      =  3000 # K
gapped_threshold   =  4500 # L
x_drop             =   900 # X
y_drop             = 15000 # Y
     A    C    G    T
A   90 -330 -236 -356
C -330  100 -318 -236
G -236 -318  100 -330
T -356 -236 -330   90
```

5.7 Availability of data and materials

The datasets supporting the conclusions of this article are available at the European Nucleotide Archive under accession number PRJEB21791. The *Arabidopsis thaliana* HiSeq read sets are available under accessions ERX2095148 and ERX2095149. The PacBio data sets are available under accessions ERX2095150 and ERX2095151. The gorilla Y chromosome assembly has been assigned accession number GCA_900199665.

5.8 Disclosure declaration

Gabino Sanchez Perez and Henri van de Geest moved to Genetwister Technologies BV after contributing to this research. Genetwister technologies BV is not affiliated with this research and is not financially or otherwise linked to the project. The authors declare that they have no competing interests.

5.9 Acknowledgments

We would like to thank Jan-Peter Nap (Hanze University of Applied Sciences, Groningen) for his comments and suggestions.

5.10 References

1. Buermans HPJ, den Dunnen JT. Next generation sequencing technology: Advances and applications. Biochim. Biophys. Acta - Mol. Basis Dis. 2014;1842:1932–41.
2. Shapiro E, Biezuner T, Linnarsson S. Single-cell sequencing-based technologies will revolutionize whole-organism science. Nat. Rev. Genet. 2013;14:618–30.
3. Gawad C, Koh W, Quake SR. Single-cell genome sequencing: current state of the science. Nat. Rev. Genet. 2016;17:175–88.
4. Tomaszkiewicz M, Rangavittal S, Cechova M, Sanchez RC, Fescemyer HW, Harris R, et al. A time- and cost-effective strategy to sequence mammalian Y Chromosomes: an application to the *de novo* assembly of gorilla Y. Genome Res. 2016;26:530–40.
5. Czyz ZT, Kirsch S, Polzer B. Principles of whole-genome amplification. Methods Mol. Biol. 2015;1347:1–14.
6. Lasken RS, Stockwell TB. Mechanism of chimera formation during the Multiple Displacement Amplification reaction. BMC Biotechnol. 2007;7:19.
7. Sabina J, Leamon JH. Bias in whole genome Amplification: causes and considerations. Methods Mol. Biol. 2015;1347:15–41.
8. Zhang K, Martiny AC, Reppas NB, Barry KW, Malek J, Chisholm SW, et al. Sequencing genomes from single cells by polymerase cloning. Nat. Biotechnol. 2006;24:680.
9. Edgar RC, Haas BJ, Clemente JC, Quince C, Knight R. UCHIME improves sensitivity and speed of chimera detection. Bioinformatics. 2011;27:2194–200.
10. Rhoads A, Au KF. PacBio sequencing and its applications. Genomics, Proteomics & Bioinformatics. 2015;13:278–89.
11. Jain M, Olsen HE, Paten B, Akeson M. The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. Genome Biol. 2016;17:239.
12. Hackl T, Hedrich R, Schultz J, Förster F. proovread: large-scale high-accuracy PacBio correction through iterative short read consensus. Bioinformatics. Oxford University Press; 2014;30:3004–11.
13. Koren S, Schatz MC, Walenz BP, Martin J, Howard JT, Ganapathy G, et al. Hybrid error correction and *de novo* assembly of single-molecule sequencing reads. Nat. Biotechnol. Nature Research; 2012;30:693–700.
14. Lee H, Gurtowski J, Yoo S, Marcus S, McCombie WR, Schatz M. Error correction and assembly complexity of single molecule sequencing reads. bioRxiv. Cold Spring Harbor Labs Journals; 2014;006395.
15. Warris S, Yalcin F, Jackson KJL, Nap JP. Flexible, Fast and Accurate Sequence Alignment Profiling on GPGPU with PaSWAS. Zhang M, editor. PLoS One. 2015;10:e0122524.
16. Warris S, Timal NRN, Kempenaar M, Poortinga AM, van de Geest H, Varbanescu AL, et al. pyPaSWAS: Python-based multi-core CPU and GPU sequence alignment. PLoS One. 2018;13.
17. Human Genome Sequencing Consortium International. Finishing the euchromatic sequence of the human genome. Nature. 2004;431:931–45.
18. Rangavittal S, Harris RS, Cechova M, Tomaszkiewicz M, Chikhi R, Makova KD, et al. RecoverY: K-mer based read classification for Y-chromosome specific sequencing and assembly. Bioinformatics. 2017;
19. Chin C-S, Alexander DH, Marks P, Klammer AA, Drake J, Heiner C, et al. Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. Nat. Methods. 2013;10:563–9.
20. Berlin K, Koren S, Chin C-S, Drake JP, Landolin JM, Phillippy AM, et al. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nat. Biotechnol. 2015;33:623–30.
21. Schmuths H, Meister A, Horres R, Bachmann K. Genome size variation among accessions of *Arabidopsis thaliana*. Ann. Bot. 2004;93:317–21.
22. Koren S, Walenz BP, Berlin K, Miller JR, Bergman NH, Phillippy AM. Canu: scalable and accurate long-read

assembly via adaptive k-mer weighting and repeat separation. Genome Res. 2017;27:722–36.

23. Cortez D, Marin R, Toledo-Flores D, Froidevaux L, Liechti A, Waters PD, et al. Origins and functional evolution of Y chromosomes across mammals. Nature. Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.; 2014;508:488–93.
24. Wu TD, Watanabe CK. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. Bioinformatics. 2005;21:1859–75.
25. Qian Z, Adhya S. DNA repeat sequences: diversity and versatility of functions. Curr. Genet. 2017;63:411–6.
26. Inagaki H, Kato T, Tsutsumi M, Ouchi Y, Ohye T, Kurahashi H. Palindrome-mediated translocations in humans: a new mechanistic model for gross chromosomal rearrangements. Front. Genet. 2016;7:125.
27. Smit AFA, Hubley R, Green P. RepeatMasker [Internet]. Available from: http://repeatmasker.org
28. Skaletsky H, Kuroda-Kawaguchi T, Minx PJ, Cordum HS, Hillier L, Brown LG, et al. The male-specific region of the human Y chromosome is a mosaic of discrete sequence classes. Nature. 2003;423:825–37.
29. Boetzer M, Henkel C V, Jansen HJ, Butler D, Pirovano W. Scaffolding pre-assembled contigs using SSPACE. Bioinformatics. 2011;27:578–9.
30. Walker BJ, Abeel T, Shea T, Priest M, Abouelliel A, Sakthikumar S, et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. PLoS One. 2014;9.
31. Nakagawa H, Fujita M. Whole genome sequencing analysis for cancer genomics and precision medicine. Cancer Sci. 2018;109:513–22.
32. Kosicki M, Tomberg K, Bradley A. Repair of double-strand breaks induced by CRISPR–Cas9 leads to large deletions and complex rearrangements. Nat. Biotechnol. 2018;
33. Warris S, Timal R. pyPaSWAS [Internet]. Available from: https://doi.org/10.5281/zenodo.51155
34. Cock PJA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. 2009;25:1422–3.
35. Jones E, Oliphant T, Peterson P. SciPy: Open source scientific tools for Python [Internet]. Available from: http://www.scipy.org
36. Schouten HJ, van de Geest H, Papadimitriou S, Bemer M, Schaart JG, Smulders MJM, et al. Re-sequencing transgenic plants revealed rearrangements at T-DNA inserts, and integration of a short T-DNA fragment, but no increase of small mutations elsewhere. Plant Cell Rep. 2017;36:493–504.
37. Chaisson MJ, Tesler G. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. BMC Bioinformatics. 2012;13:238.
38. Ye C, Hill C, Ruan J, Zhanshan, Ma. DBG2OLC: Efficient assembly of large genomes using the compressed overlap graph. 2014;
39. Ye C, Ma ZS, Cannon CH, Pop M, Yu DW. Exploiting sparseness in *de novo* genome assembly. BMC Bioinformatics. 2012;13 Suppl 6:S1.
40. Gurevich A, Saveliev V, Vyahhi N, Tesler G. QUAST: quality assessment tool for genome assemblies. Bioinformatics. 2013;29:1072–5.
41. Smit A, Hubley R. rmblast [Internet]. Available from: http://www.repeatmasker.org/RMBlast.html
42. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, et al. BLAST+: architecture and applications. BMC Bioinformatics. 2009;10:421.
43. Bao W, Kojima KK, Kohany O. Repbase Update, a database of repetitive elements in eukaryotic genomes. Mob. DNA. BioMed Central; 2015;6:11.
44. Harris RS. Improved pairwise alignment of genomic DNA. Thesis, Pennsylvania State University. 2007;
45. Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. 2013;
46. Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 2010;26:841–2.

5.11 Supplementary materials

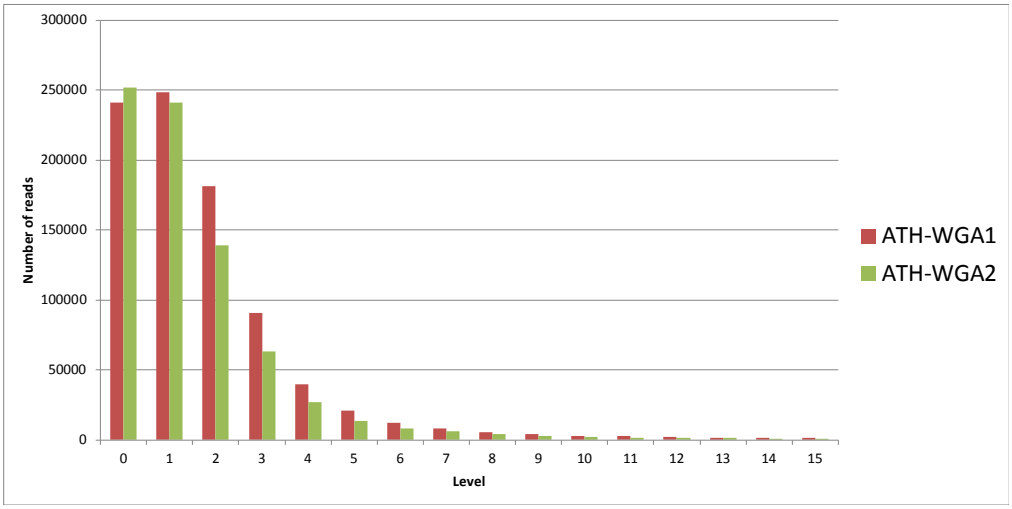

Figure 5.5. S1. Number of Pacasus iterations per read in the cleaned data set. Level 0 indicates that the original read had no detectable palindromic sequence. level 1 indicates the read was processed only once, level 2 twice, etcetera.

6 Mining functional annotations across species

Sven Warris, Bart van de Vossenberg, Steven Dijkxhoorn, Teije van Sloten, Theo van der Lee, Jan-Peter Nap, Dick de Ridder

Published in part as Bart van de Vossenberg & Sven Warris, *et al.*, ***Comparative genomics of chytrid fungi reveal insights into the obligate biotrophic and pathogenic lifestyle of*** **Synchytrium endobioticum**, Nature Scientific Reports, https://doi.org/10.1038/s41598-019-45128-9, 2019

6.1 Abstract

Background

Numerous tools and databases exist to annotate and interpret the functions encoded in a genome (InterProScan, KEGG, Gene Ontology (GO) etc.). However, analyzing and comparing functionality across several genomes is not trivial, whereas such comparative analyses are essential for identifying genes related to a particular biological function.

Results

We present a novel approach for comparative functional analyses in which KEGG, GO, CAZyme and InterProScan results from several species are collected and stored in a Neo4j graph database: Multi-Species Whole Annotation System Database (MuSWAS db). Using a developed plug-in for Cytoscape to connect to Neo4j, the database can be queried and visualized. The visualization of functional annotation (sub)graphs facilitates comparisons and grouping of functional annotation across species. The approach is presented in the context of the functional analysis of the fungal species *Synchytrium endobioticum*, the causal agent of potato wart disease, compared to other fungal species with comparable taxonomy or lifestyle.

Conclusions

The combination of the functional annotations of different species in a single graph database and the ability to visualize and identify overlaps and differences through scripts and in Cytoscape generates novel insights in functions of genome components of a single species or a group of organisms. The approach we present is generic and suitable for a large number of species. The developed plug-in supports the analyses of any graph stored in Neo4j. The analyses of several fungal genomes identify genes that are likely to play a role in the obligate biotrophic and pathogenic lifestyle of *Synchytrium endobioticum*.

6.2 Background

A first step following *in silico* reconstruction of a genome [1] is usually to produce a structural annotation of the genome to identify repeats [2], genes [3,4] and other genomic features [5]. In combination with sequence information, these data are used for comparative genomics efforts to interpret the functional make-up of the organism under study, often compared to that of a related organism. Genes are functionally annotated using (a) InterProScan [6] or Blast2GO [7], (b) analyses of the presence or absence of pathway elements in the Kyoto Encyclopedia of Genes and Genomes (KEGG) [8,9], and (c) Gene Ontology (GO) term enrichment [10]. These approaches largely focus on comparing a single species of interest to a single known reference species. In our research, we aim to find overlap and difference between groups of species on a

functional level. To the best of our knowledge however, no tools are currently available to store, visualize and analyze the functional annotation of several species combined. Both GO and KEGG databases contain highly connected information, suitable for storing in a graph database [11] and subsequently performing queries on the relationships within these datasets. InterProScan results can be added, and the resulting graph can be queried to perform analyses on the integrated data for complex relationships. Additionally we performed an analysis of carbohydrate-active enzymes (CAZymes), as the ability to use different types of carbohydrates is an important feature of plant pathogens [12], and added these results to the graph database. Here, we present a new approach building on such graph database technology [13] and Cytoscape [14] to mine functional annotations of several species. It enables comparison of for example GO terms and KEGG pathways and on-the-fly visualization of the results [15] across multiple species.

The approach is presented in the context of the comparative functional analyses of the genome of the fungal species *Synchytrium endobioticum*. This fungus is the causal agent of potato wart disease, or black scab, currently one of the most important quarantine diseases of cultivated potato [16]. *Synchytrium* is the type genus of the family Synchytiaceae, order Chytridiomycetes, division Chytridiomycota, also called chytrids, a basal lineage of fungi (lower fungi) with motile zoospores. The genus *Synchytrium* contains over 200 species, most of which are obligate pathogens on flowering plants, ferns, mosses or algae, although a saprobic free-living *Synchytrium* species, *S. microbalum*, has also been reported [17]. *Synchytrium endobioticum* is by far the best-studied species of the genus, yet little is still known about the molecular mechanisms of its obligate biotrophic and pathogenic lifestyle. More insight into the pathogenicity mechanisms of *S. endobioticum* is likely to come from comparative functional genomics of different fungi with similar or contrasting lifestyles. Thus far, only a few genomes of species from the chytrid phylum have been sequenced and studied. Therefore, to gain insights into the lifestyle of *S. endobioticum* we compared the genomes of two isolates with those of nine other chytrids and six so-called higher fungi (members of the Ascomycetes and Basidiomycetes) that produce hyphae or a mycelium [27] and are either obligatory biotrophs as *S. endobiotium*, or culturable as the other chytrids. For all seventeen fungal genomes, we ran InterProScan to annotate their genes with GO terms and KEGG enzymes. The analysis and determination of CAZymes encoding genes were performed using a similar methodology described in [18]. Results were combined in a Neo4j graph database, named Multi-Species Whole Annotation System Database (MuSWAS db). We aimed to identify and characterise genes linked to obligate biotrophy, cell wall degradation, pathogenicity and sexual cycle of chytrid fungi compared to non-chytrid fungi. In this chapter, we focus on the aspects of graph technology for such functional annotation of the different genomes and subsequent comparisons. Detailed descriptions of the fungal material included, as well as the full results of the integral genome comparisons of all fungi are presented elsewhere [19].

6.3 Materials and methods

6.3.1 Genome data

The data consists of the genes of seventeen fungal species [19], in total 157,709 genes. The seventeen genomes were divided into four groups: 'Chytrid, obligatory biotroph' (ChytObl) with two isolates of *S. endobiotium*; 'Chytrid, culturable' (ChytCult) with nine culturable Chytrid species; 'Control, obligate biotroph' (CtrlObl), with three obligate biotrophic non-Chytrid (higher) fungi, and 'Control, culturable' (CtrlCult), with three culturable non-Chytrid (higher) fungi.

6.3.2 Software and databases

InterProScan version 5.16-55.0 [6] was used to annotate the function of genes using the default settings. Python version 3 [20] and the BioPython package [21] were used for processing protein files, InterProScan output and the GO-basic dataset from the Gene Ontology (GO) database (May 2017) [10]. The Python modules Bio.KEGG and Bio.graphics were used to connect to the KEGG API [22] and process the results. The Python package Neo4j V1 [23] was used to connect to the Neo4j database [11]. Queries to the database were written in the Neo4j Cypher query language version 3.4+ [24]. CAZyme [25] information was retrieved from the web site http://www.cazy.org/.

The Cytoscape plug-in was developed for Cytoscape version 3.6+ [14] and was written in Java 1.8. The already available cyNeo4j plug-in [26] was used as a starting point for the newly developed plug-in.

6.4 Results

6.4.1 Gene Ontology data

To process InterProScan results for GO terms, the GO-basic data set was downloaded from the Gene Ontology Consortium (GOC) website (http://geneontology.org/) and stored in MuSWAS db. Each node in the graph corresponds to a GO term, and the "ISA" edges correspond to the is-a relationships as defined by the GOC. Figure 6.1 shows an example of the top-level node 'biological process' and some of its descendants based on these relationships. It is important to note that the GO data set does not follow a tree structure, but rather constitutes a general Directed Acyclic Graph (DAG), i.e. many GO terms have more than one parent. With the InterProScan output, the number of times a particular GO term is found in a given species is determined and added as an attribute to the node representing that particular GO term. As a result, each GO term node includes the total number of times it has been assigned to a particular organism. Counts per species group are determined by summing individual species counts. By propagating the counts from the leaves to the three top-level GO terms "biological process", "molecular function" and

"cellular component", each level in the GO graph contains the number of times that GO term or its descendants are found by InterProScan. As the GO graph is a DAG, a descendant can be linked more than once to a level; yet species count values should only be used once otherwise the GO term assignments will be overrepresented in the higher-level terms. To prevent double counting a query in Cypher was used to select only distinct GO terms below a given level in the GO graph.

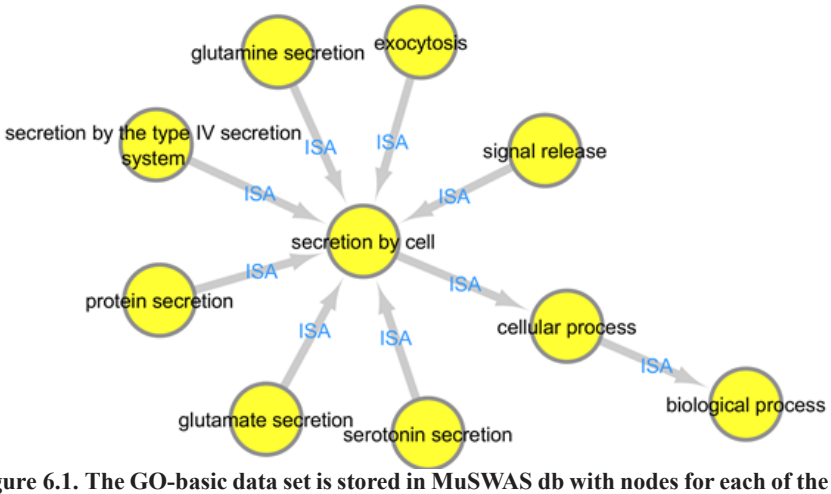

Figure 6.1. The GO-basic data set is stored in MuSWAS db with nodes for each of the GO terms connected to the more generic term using the ISA relationship. Part of this GO term DAG showing the top-level node 'biological process' and some of its descendants. GO terms are depicted using yellow circles and the ISA relationships are shown here as light grey arrows.

6.4.2 KEGG pathways

To link the enzyme annotation to KEGG pathways, pathway identifiers and Enzyme Commission (EC) numbers were extracted from the InterProScan output. Using the KEGG API, the structures of the pathways are downloaded in XML (KGML) format and stored in the MuSWAS db. Pathway elements are directly translated into nodes ('enzyme', 'pathway', 'map', etc.) and relationships ('ECrel', 'maplink', 'reaction' etc.). Enzymes are linked to a pathway using the 'in' relationship. An enzyme can occur in more than one pathway and can, therefore, have more than one outgoing 'in' relationship. The XML data also contains graphical properties to place enzymes on the KEGG pathway images. These are set as attributes of the 'in' relationship to establish unique enzyme-pathway links. Relationships between an enzyme and its substrates and products are not represented in the XML data. Therefore, to complete the pathway graph, for each enzyme the KEGG annotation is retrieved through the KEGG API and the relationships 'usedby' and 'produces' are created to connect enzymes with compounds. The result is a Directed Cyclic Graph (DCG). In Figure 6.2, an example of such a DCG is shown.

6.4.3 Connecting KEGG and GO to CAZy

The cell wall degradation pathway of *S. endobioticum* is likely catalysed by carbohydrate-active enzymes (CAZymes), large families of enzymes that degrade, modify, or create glycosidic bonds [25] as documented in the CAZy database [34]. In all fungal genomes, CAZymes were predicted using the on-line tool dbCAN [33] using default parameters. The number of times a particular CAZyme was found in a genome was determined and subsequently normalized by dividing the counts by the total number of proteins in present in the genome. These data were added to MuSWAS db. Links from CAZymes to GO terms were retrieved from the CAZy website and also added to MuSWAS db.

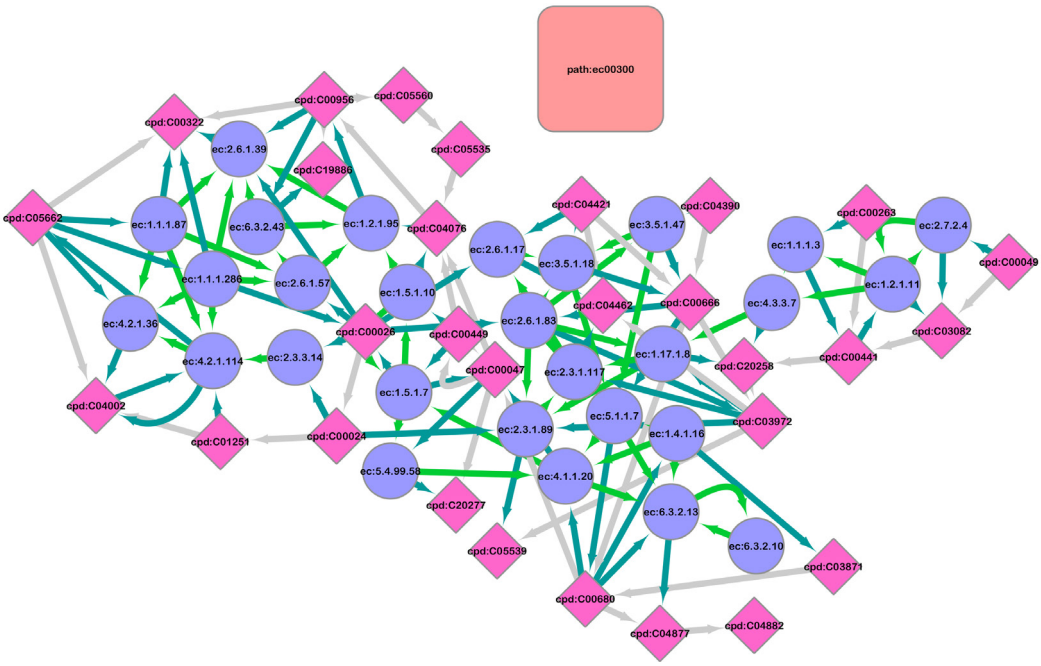

Figure 6.2. Structure of the KEGG pathway graph as DCG. The small KEGG pathway ec:00300 ("Lysine biosynthesis", pink rounded box) with its enzymes (blue circles) and compounds (pink diamonds) is shown as an example of a DCG. For clarity of presentation, the 'in' relationships from the compound and enzyme nodes to the pathway node are not shown, which would otherwise connect all nodes to the pathway node. Relationships shown are the 'usedby' and 'produces' edges between enzymes and compounds (blue), the 'ECrel' relationships between enzymes (green) and the 'reaction' relationship between compounds (grey). Shapes and colors were set by Cytoscape styles.

For each of the seventeen organisms, the number of times an enzyme is predicted by InterProScan is added to the particular enzyme node. Similar as for the GO terms, counts are aggregated for species groups and added to the enzyme nodes.

6.4.4 Connecting KEGG to GO

The on-line KEGG database contains links to GO terms for many of its enzymes. Using the same data as for connecting compounds to enzymes, these GO terms are linked to the enzymes with the 'crossConnect' relationship (Figure 6.3). This linkage allows for visualization of these otherwise implicit relationships and, more importantly, shows further evidence of additional or missing functions: when both enzyme and associated GO terms are missing in one group of species compared to another group, the likelihood of a false negative result decreases.

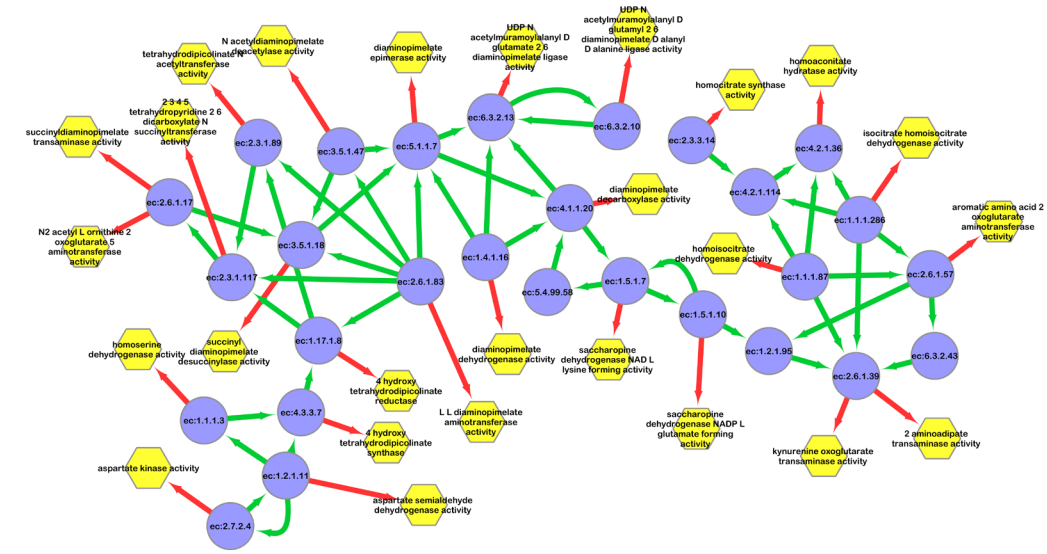

Figure 6.3. Linking KEGG enzymes to GO terms. KEGG enzymes (blue circles) and GO terms (yellow hexagons) are linked through the 'crossConnect' relationship (red edges). The green edges are the 'ECrel' relationships between the different enzymes. Again, for clarity, not all relationships and nodes stored in MuSWAS db are shown. Shapes and colours were set by Cytoscape styles.

6.4.5 Overview of the data in MuSWAS db

With the collected and aggregated data, the resulting graph database of the fungal species considered contains a total of 54,684 nodes (Table 6.1) and 162,056 relationships (Table 6.2). With inclusion of the functional annotation from InterProScan, 4,872 new relationships (crossConnect) were established (Table 6.2). The next step was to develop appropriate ways of mining these data, for the various groups of fungi that can be distinguished (Table 6.3).

6.4.6 Cytoscape and the Neo4j plug-in

Cytoscape is a powerful graph visualization tool with many built-in features and many external tools available through the plug-in app store [12]. Within Cytoscape, it is easy to color and shape ('style') nodes and edges based on the information in a graph. Also, information can be linked to external websites such as NCBI. This makes Cytoscape [14] a popular tool for the visualization and analyses of biological networks, such as protein-protein interaction networks [27] or gene co-expression networks [28]. However, loading all data stored in our Neo4j database (Tables 6.1 and 6.2) in Cytoscape is impossible on a standard desktop computer due to memory constraints. To visualize subgraphs and query them within Cytoscape, it needs a connection to the Neo4j database. Therefore, a Java implementation of a basic yet outdated Neo4j Cytoscape plug-in, cyNeo4j [26,29], was refactored and extended to connect Cytoscape directly with a Neo4j database [11]. The plug-in allows for querying the database and storing created networks within Cytoscape, and for storing networks imported from other sources in Cytoscape in the database. The plug-in also can read an XML file with predefined Cypher queries which allows for storing important and/or often used queries on disk as templates for future use. The network resulting from running such a query is then visualized in Cytoscape. Additional features of the plug-in include interactively expanding and connecting nodes in Cytoscape based on the graph structure in the database. These features are independent of the structure of the graph, as the database is queried for the nodes and relationships available for the selected node(s). A more descriptive list of features is given in Supp. Materials.

Table 6.1. Number of nodes in MuSWAS db after processing the functional annotation from InterProScan.

Node	Number
GO term	47,059
Enzyme	3,019
Pathway	114
Map	164
Compound	4,328
Total	**54,684**

Table 6.2. Number of relationships in MuSWAS db after processing the functional annotation from InterProScan.

Relationship	Number
crossConnect	4,872
isa	71,400
is	228
usedby	11,248
maplink	5,300
ECrel	25,878
in	22,948
produces	12,150
reaction	8,032
Total	**162,056**

Table 6.3. Subgrouping of fungal species for comparative analyses.

Group name (number of genomes)	Coloring	ChytObl (2)	ChytCult (9)	CtrlCult (3)	CtrlObl (3)
All (17)	Green	+	+	+	+
Only in culturables (12)	Orange	-	+	+	-
Only in obl. (5)	Blue	+	-	-	+
Absent in obl. Chytrids (15)	Purple	-	+	+	+
Only in obl. Chytrids (2)	Red	+	-	-	-
Higher fungi (6)	Silver	-	-	+	+
Unclassified	Yellow	-	-	-	-
Other	Pink				

Grouping based on present (+) or absent (-) of enzymes, GO terms, etc. in one of the four main groups used for a more detailed comparison of the different species based on their lifestyle. 'Unclassified' in this context means that the element (e.g.. an enzyme) was not predicted to be present in any of the genomes and 'Other' means that the combination of genomes containing the predicted element does not fit in any of the other groups.

6.4.7 Detection of differences between species

With Cytoscape and the newly developed Neo4j plug-in, the graph database is mined for the functional annotation of the individual species and the four specified groups. For the comparison of multiple species, a first step is to subclassify them: which elements are found in all species, which are specific for the culturable species, for the obligatory biotrophs, obligatory biotrophic chytrids and so on (Table 6.3). Such multi-species comparisons allow inferences on components of lifestyle. This information is extracted from MuSWAS db with dedicated Cypher queries and Python scripts. Each of these classes is then assigned a different color with the style filters of Cytoscape. Examples of the classifications outlined in Table 6.3, as visualized in Cytoscape after extraction of the data from MuSWAS db, are shown in Figures 6.5-6.7. Figure 6.4 shows the compounds connected to and the GO terms associated with four example enzymes. Node colors

indicate to which of the classes each element in the graph belongs and show that these four enzymes and their associated GO terms are not predicted to be present in this species group. These results indicate that the obligatory biotrophic chytrids are missing four enzymes in the "Glyoxylate and dicarboxylate metabolism" pathway (path:ec00630). If confirmed, these missing enzymes could be of interest to investigate their role in the obligatory biotrophic lifestyle of *S. endobioticum*. Many similar biological inferences on lifestyle and other characteristics of fungi can be made by comparisons based on the data in the resulting fungal Neo4j database; a number of results are presented elsewhere [17].

As a second example, an inspection of the GO terms related to motility, such as "motile cilium" and "cilium", showed these were indeed only detected in the chytrid species which are characterized by motile zoospores rather than hyphae or mycelium (Figure 6.5).

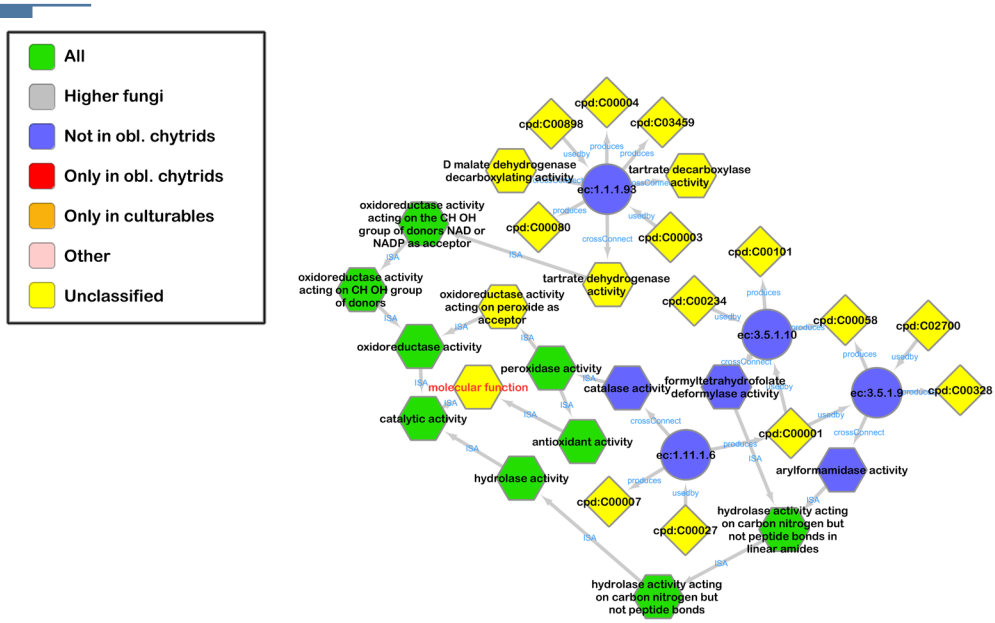

Figure 6.4. Data in MuSWAS db showed that four enzymes (blue circles) of pathway ec:00630 ("Glyoxylate and dicarboxylate metabolism") are missing in the two obligate biotrophs (i.e. the two *S. endobioticum* isolates). The compounds associated with the enzymes are visualized by diamond shaped nodes. GO terms linked to these enzymes by the 'crossConnect'-edge are shown as hexagons with the paths to the top-level node 'molecular function' (text in red). Other edges shown are the GO 'ISA' relationship to connect the GO terms and the KEGG pathway relationships 'produces' and 'usedby' to connect enzymes with compounds.

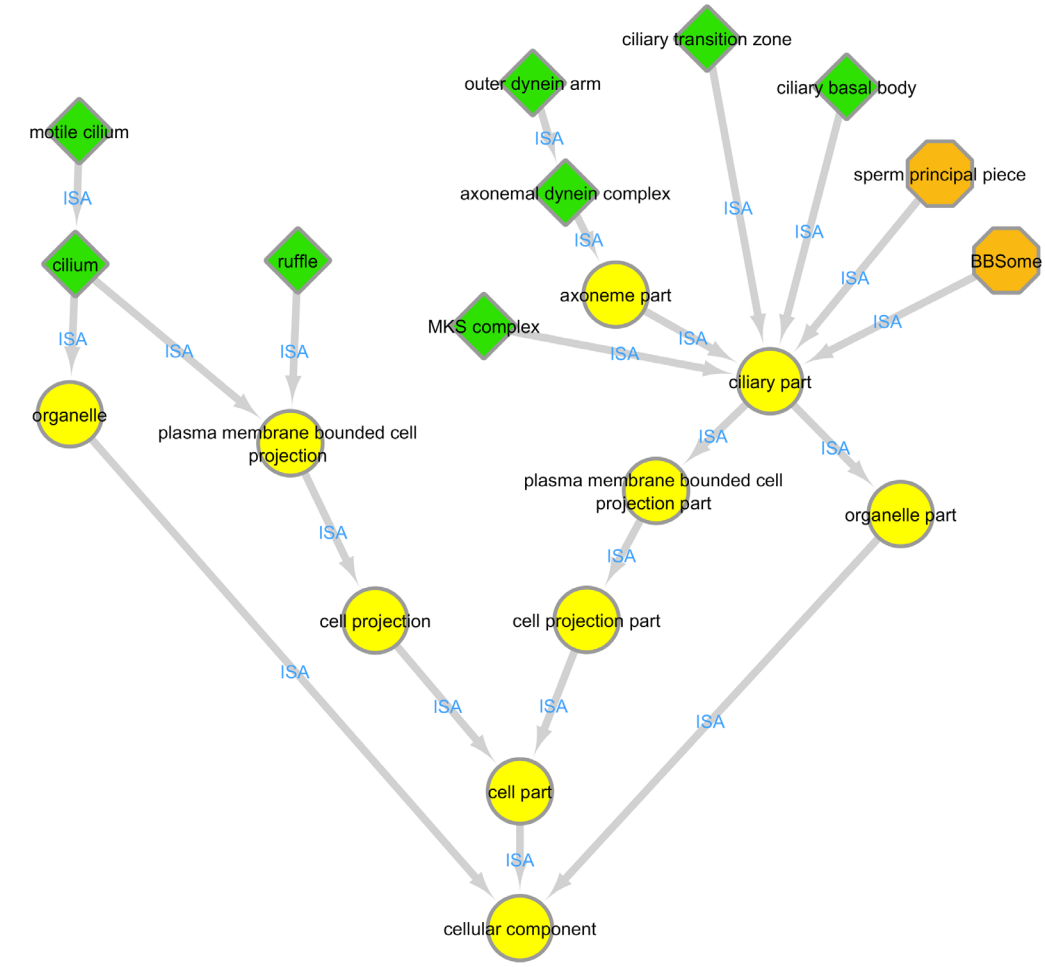

Figure 6.5. GO-terms associated with flagella and movement, and their occurrences in chytrid species and higher fungi. Shapes in the network indicate the different GO-terms, with "Cellular Component" as the highest-level term. Grey arrows indicate the "ISA" relationship between the GO-terms as defined in the Gene Ontology database. The different shapes indicate if a given GO-term is present in culturable and obligate biotrophic chytrid species but not in higher fungi (green diamond), present in one or more culturable chytrid species (orange hexagon), or unclassified (yellow circles). Shapes and colors are set by using the styles in Cytoscape and are filtered based on the information in the graph database.

6.4.8 Accuracy and consistency

The results presented here are based on InterProScan analyses. As InterProScan is a predictive method, false positive (FP) and false negative (FN) predictions are expected to be found. To investigate the impact of these FP/FN predictions, the KEGG pathway analysis was benchmarked using the six control species with known biochemical reference pathways included in the KEGG database. Overall accuracy (84.5%) was mainly influenced by false negative results (Supplementary Report). Such a case is illustrated by aconitate hydratase (EC:4.2.1.3), an enzyme from the citrate cycle, which was detected in all genomes analyzed using InterProScan but was not colored in the KEGG pathway because of a missing link in the KEGG database (Figure 6.6). In case a functional element appears to be missing in a genome, additional independent methods should be used to confirm these findings. The high accuracy implies a low number of false positives in the control species, so a particular function predicted in a genome is likely to be true .

Combining the KEGG, GO and CAZyme annotations gave new insights into the lifestyle of *S. endobioticum*. However, during our analyses discrepancies between different online resources became apparent, hampering further research. KEGG, GO and CAZymes databases are data sources which contain redundant information: KEGG contains which GO terms are associated with enzymes and visa-versa and CAZymes are linked to enzymes and these data should be the same across the three databases. We found several discrepancies which, at first glance, gave inconclusive results. Manual curation was required to correct these findings [19]. These results show that links of terms and features are not necessarily consistent between the different databases combined in MuSWAS db. By adding the information of the three databases in a single data source (MuSWAS db) discrepancies are more apparent and hence can be corrected in the original sources.

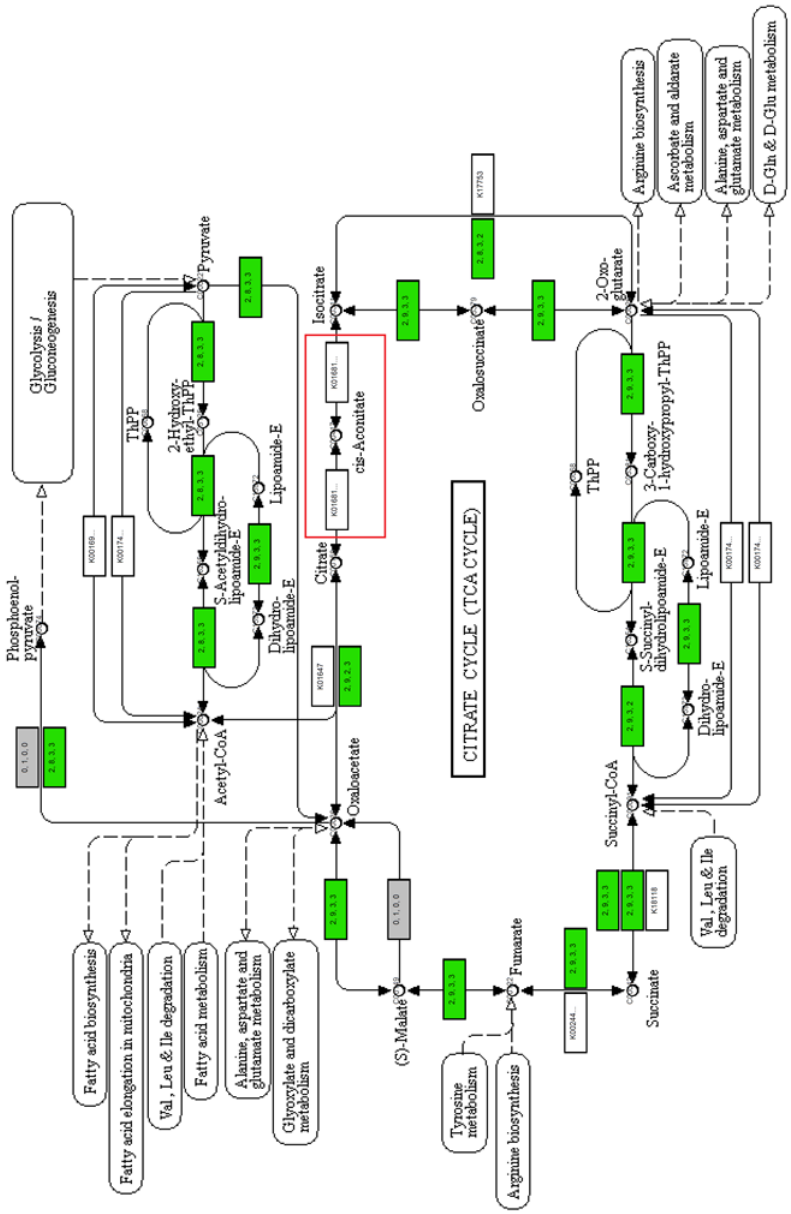

Figure 6.6. KEGG reference pathway 20: citrate cycle (TCA). Numbers (a, b, c, d) replacing the KEGG Orthology (KO) numbers in the enzymatic steps indicate the number of isolates in a given group for which the corresponding gene was detected. a: obligate biotrophic chytrids (maximum number of genomes = 2); b: culturable chytrids (max = 9); c: culturable higher fungi (max = 3); and d: obligate biotrophic higher fungi (max = 3). The maximum score that can be obtained is "2,9,3,3", which is the case for all green boxes. White boxes are unassigned enzymes. Boxed in red is the enzyme aconitate hydratase (EC:4.2.1.3), which should be found in all genomes. The green colored enzymes show that all required enzymes for the TCA cycle except aconitate hydratase were found in all genomes.

6.5 Discussion and conclusions

We have demonstrated how the combination of a Neo4j graph database and Cypher with Cytoscape enables comparative analyses of GO, KEGG, CaZy and InterProScan data across multiple genomes that would otherwise be hard and time-consuming. Genomes can easily be grouped based on any criterion set, allowing comparisons at different levels of biology. In the example dataset of *S. endobioticum* with other chytrid and non-chytrid fungal species, various aspects of fungal lifestyle and the pathogenicity of *S. endobioticum* on potato have given leads for new research and insights in biology described in more detail elsewhere [17]. The connection with Cytoscape adds intuitive and easy visualization to the analyses. The developed Cytoscape plug-in is generic: other publicly available databases such as the Reactome [30] pathway database or any other Neo4j database can also be queried with the plug-in. The plug-in allows for sending any valid query to the connected database, but these queries still need to be designed by the user. These queries can become long and complex quickly, hence a good understanding of Neo4j and Cypher is still needed. Fortunately the plug-in allows for storing and re-using of queries, allowing for sharing these queries with others.

A possible additional application of the approach is in metagenomics. In environments, such as soil, natural waters and gastric systems [31–35], communities of (micro-)organisms collaborate or compete for space and food. The first step in any metagenomics analysis is to functionally annotate all genes found in the DNA or RNA data [36–38]. With the approach shown here, this results in a graph database allowing to identify which pathways are present, which compounds are produced and which biological processes occur. It would not be limited to a single metagenomics sample: by grouping annotations, it will be possible to perform easy comparative analyses of different samples. Especially when combined with the annotation of individual microbial genomes, functional analyses of metagenome data is likely to get a huge boost.

For simplicity, the GO-basic dataset (with only “ISA” relationships) was used, making the graph less complex than when using the full GO-plus dataset. This selection is not an intrinsic limitation: when the GO-plus dataset is more appropriate, it can be used in the same way, although the resulting relationships are likely to become more difficult to interpret.

Results indicate that the obligatory biotrophic chytrids miss four enzymes in the “Glyoxylate and dicarboxylate metabolism” pathway (path:ec00630), potentially of importance in the obligatory biotrophic lifestyle of *S. endobioticum*. However, caution is advised: although the GO term ‘catalase activity’ is not predicted for this group of fungi, the higher-level GO term ‘peroxidase activity’ is predicted to be present in all species groups, so other enzymes may take over. Also, the detailed analyses of the cellulase activity (CAZymes) show that links of terms and features are not necessarily consistent between databases combined in MuSWAS. Therefore,

relying on a single data source is not recommended, and predictions based on multiple sources of data continue to need scrutiny and preferably independent validation: data can be contradictory, inconclusive or incorrect. Adding information from different sources and linking them through known relationship in MuSWAS db will make these irregularities more obvious. Such results should be taken as motivation to contact database managers to improve the consistency and interoperability of biological data and databases. The downside of combining these data in a single data source is that updates in the original sources (such as corrections in the links) are not automatically updated in MuSWAS db and this update process would require a rerun of (parts of) the analyses. Adding additional genomes to an existing data set in MuSWAS db requires running InterProScan and CAzyme analyses on these genomes and adding the resulting counts to MuSWAS db. Aggregated counts then need to be recalculated based on this new information and from that point on data can be visualized again. Main bottlenecks in adding more genomes are therefore the genome annotation analytics (in CPU time) and visualization plus interpreting the results (in time spent by the researcher).

If such issues are taken into account, the concept and use of graph databases presents considerable added value to biological analyses and is likely to get more attention. An increasing number of bioinformatics tools, including sequence alignment [39], support Neo4j databases, creating more demand for user-friendly graph visualization. Our focus on the functional annotation of genes did not take genome structure into account. Pan-genome methods [40], such as PanTools [41], allow storing genome structures in a graph database. The combination of the approach presented here with a pan-genome graph will allow new ways of querying and visualizing data across multiple genomes. This combination will make it possible, for example, to retrieve the locations of genes in a pathway of interest and assess functional sequence variation much easier and faster than currently available.

6.6 Authors' contributions

SW wrote all Python scripts, performed the Interproscan data analysis, designed the cypher queries and created the plots. SW, SD and TvS redesigned, refactored and extended the Neo4j plugin for Cytoscape. BvdV supplied the biological data, created the groups of species and designed the biological experiments. SW and BvdV made the biological interpretations of the results. SW wrote the manuscript and edited the manuscript after comments by DdR and JPN. Both DdR and JPN supervised SW during this research.

6.7 Acknowledgements

Part of the research was funded through the Big Data strategic project at Wageningen UR. Ordina provided time and resources through its JTech research program. We thank Ordina and Genetwister allow SD and TvS to contribute to the Open Source plug-in in company time, without these companies having any financial, legal or other benefits from this research, nor any say in the content of the project.

6.8 References

1. Koren S, Walenz BP, Berlin K, Miller JR, Bergman NH, Phillippy AM. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Genome Res. 2017;27:722–36.
2. Qian Z, Adhya S. DNA repeat sequences: diversity and versatility of functions. Curr. Genet. 2017;63:411–6.
3. Hoff KJ, Lange S, Lomsadze A, Borodovsky M, Stanke M. BRAKER1: unsupervised RNA-Seq-based genome annotation with GeneMark-ET and AUGUSTUS. Bioinformatics. 2015;32:767–9.
4. Holt C, Yandell M. MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. BMC Bioinformatics. 2011;12:491.
5. Warris S, Boymans S, Muiser I, Noback M, Krijnen W, Nap J-P. Fast selection of miRNA candidates based on large-scale pre-computed MFE sets of randomized sequences. BMC Res. Notes. 2014;7:34.
6. Zdobnov EM, Apweiler R. InterProScan - an integration platform for the signature-recognition methods in InterPro. Bioinformatics. 2001;17:847–8.
7. Gotz S, Garcia-Gomez JM, Terol J, Williams TD, Nagaraj SH, Nueda MJ, et al. High-throughput functional annotation and data mining with the Blast2GO suite. Nucleic Acids Res. 2008;36:3420–35.
8. Kanehisa M, Goto S. KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Res. 2000;28:27–30.
9. de Vries RP, Riley R, Wiebenga A, Aguilar-Osorio G, Amillis S, Uchima CA, et al. Comparative genomics reveals high biological diversity and specific adaptations in the industrially and medically important fungal genus *Aspergillus*. Genome Biol. 2017;18:28.
10. Carbon S, Ireland A, Mungall CJ, Shu S, Marshall B, Lewis S. AmiGO: online access to ontology and annotation data. Bioinformatics. 2009;25:288–9.
11. Neo4J [Internet]. [cited 2016 Sep 1]. Available from: https://neo4j.com/
12. Amselem J, Cuomo CA, Van Kan JAL, Viaud M, Benito EP, Couloux A, et al. Genomic analysis of the necrotrophic fungal pathogens *Sclerotinia sclerotiorum* and *Botrytis cinerea*. PLoS Genet. 2011;7:e1002230.
13. Miller JJ. Graph database applications and concepts with Neo4j. Proc. South. Assoc. Inf. Syst. Conf. Atlanta, GA, USA. 2013.
14. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. 2003;13:2498–504.
15. Dijkxhoorn S, Sloten T van, Warris S. Cytoscape Neo4J Plugin [Internet]. 2017. Available from: https://github.com/corwur/cytoscapeneo4j
16. Smith IM, Burger B. Quarantine pests for Europe. C.A.B. International; 1997.
17. Longcore JE, Simmons DR, Letcher PM. *Synchytrium microbalum sp. nov*. is a saprobic species in a lineage of parasites. Fungal Biol. 2016;120:1156–64.
18. Zerillo MM, Adhikari BN, Hamilton JP, Buell CR, Lévesque CA, Tisserat N. Carbohydrate-Active Enzymes in pythium and their role in plant cell wall and storage polysaccharide degradation. PLoS One. 2013;8:e72572.
19. Vossenberg BTLH van de, Warris S, Nguyen HDT, Gent-Pelzer MPE van, Joly DL, Geest HC van de, et al. Comparative genomics of chytrid fungi reveal insights into the obligate biotrophic and pathogenic lifestyle of *Synchytrium endobioticum*. Sci. Rep. 2019;9.
20. Python [Internet]. http://www.python.org. Available from: http://www.python.org
21. Cock PJA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. 2009;25:1422–3.

22. Kawashima S, Katayama T, Sato Y, Kanehisa M. KEGG API: a web service using SOAP/WSDL to access the KEGG system. Genome Informatics. 2003;14:673–4.
23. Neo4J Team. Neo4J Python driver [Internet]. Available from: https://github.com/neo4j/neo4j-python-driver
24. Neo4j. Cypher language [Internet]. Available from: https://neo4j.com/developer/cypher-query-language/
25. Lombard V, Golaconda Ramulu H, Drula E, Coutinho PM, Henrissat B. The carbohydrate-active enzymes database (CAZy) in 2013. Nucleic Acids Res. 2014;42:D490–5.
26. Summer G, Kelder T, Ono K, Radonjic M, Heymans S, Demchak B. cyNeo4j: connecting Neo4j and Cytoscape. Bioinformatics. 2015;31:3868–9.
27. Browne F, Wang H, Zheng H. Investigating the impact human protein–protein interaction networks have on disease-gene analysis. Int. J. Mach. Learn. Cybern. 2018;9:455–64.
28. Liu W, Li L, Long X, You W, Zhong Y, Wang M, et al. Construction and analysis of gene co-expression networks in Escherichia coli. Cells. 2018;7:19.
29. Summer G. cyNeo4j [Internet]. 2014. Available from: https://github.com/gsummer/cyNeo4j
30. Fabregat A, Korninger F, Viteri G, Sidiropoulos K, Marin-Garcia P, Ping P, et al. Reactome graph database: Efficient access to complex pathway data. PLOS Comput. Biol. 2018;14:e1005968.
31. Jansson JK, Hofmockel KS. The soil microbiome — from metagenomics to metaphenomics. Curr. Opin. Microbiol. 2018;43:162–8.
32. Cochran AT, Bauer J, Metcalf JL, Lovecka P, Sura de Jong M, Warris S, et al. Plant selenium hyperaccumulation affects rhizosphere: enhanced species richness and altered species composition. Phytobiomes. 2018;
33. Krishnan M, Bharathiraja C, Pandiarajan J, Prasanna VA, Rajendhran J, Gunasekaran P. Insect gut microbiome - An unexploited reserve for biotechnological application. Asian Pac. J. Trop. Biomed. 2014;4:S16-21.
34. Zhernakova A, Kurilshikov A, Bonder MJ, Tigchelaar EF, Schirmer M, Vatanen T, et al. Population-based metagenomics analysis reveals markers for gut microbiome composition and diversity. Science. 2016;352:565–9.
35. Kennedy J, Flemer B, Jackson SA, Lejon DPH, Morrissey JP, O'Gara F, et al. Marine metagenomics: new tools for the study and exploitation of marine microbial metabolism. Mar. Drugs. 2010;8:608–28.
36. Meyer IM. A practical guide to the art of RNA gene prediction. Brief. Bioinform. 2007;8:396–414.
37. Wang Z, Gerstein M, Snyder M. RNA-Seq: a revolutionary tool for transcriptomics. Nat. Rev. Genet. 2009;10:57–63.
38. Thudi M, Li Y, Jackson S a, May GD, Varshney RK. Current state-of-art of sequencing technologies for plant genomics research. Brief. Funct. Genomics. 2012;11:3–11.
39. Warris S, Timal NRN, Kempenaar M, Poortinga AM, van de Geest H, Varbanescu AL, et al. pyPaSWAS: Python-based multi-core CPU and GPU sequence alignment. PLoS One. 2018;13.
40. Marschall T, Marz M, Abeel T, Dijkstra L, Dutilh BE, Ghaffaari A, et al. Computational pan-genomics: status, promises and challenges. Brief. Bioinform. 2016;19:bbw089.
41. Sheikhizadeh S, Schranz ME, Akdel M, de Ridder D, Smit S. PanTools: representation, storage and exploration of pan-genomic data. Bioinformatics. 2016;32:i487–93.

6.9 Supplementary information

Features of the Cytoscape Neo4j Plug-in

The plug-in allows the user to connect to Neo4j with a username/password using the Bolt interface.

Importing graphs

There are three main methods of importing a graph:

- Import all nodes and edges from Neo4j into Cytoscape
- Import a cypher query into Cystoscape
- Import a stored query (template) into Cytoscape

Exporting networks

The plug-in allows you to export any Cytoscape Network to Neo4j. This network can be an updated version of an imported graph or a network from a different source. Nodes / relationships removed from a graph in Cytoscape will also result in these elements being removed from the Neo4j database after the export.

Expanding nodes

The plugin allows you to expand a single node, selected nodes or all nodes in the network at once. This way you can browse through your graph.

Main menu:

- Expand all (selected) nodes in the network through all edges (bidirectional)
- Expand all (selected) nodes, incoming edges only
- Expand all (selected) nodes, outgoing edges only

Context menu:

- Expand single node, bidirectional, incoming or outgoing edges
- Expand single node, bidirectional, incoming or outgoing edges, based on the available edges connected to this node
- Expand single node, bidirectional, incoming or outgoing edges, based on the available nodes connected to this node

Other features

- Show all edges (relationships) between all nodes in the network or only between selected nodes.
- Get the shortest paths from the database between the selected nodes. When more than two nodes are selected, all combinations will be queried: Neo4j does not allow shortest path calculations between more than two nodes (a.k.a. ‘via’).

6.10 Supplementary Report

This report shows the predictive value of Interproscan for KEGG pathway analyses. For this six different reference species are used.

Definitions

Positive: number of positive elements in the pathway: these elements are present in the reference species (P)

Negative: number of negative elements in the pathway: these elements are not present in the reference species (N)

True positive (TP): element in the reference pathway correctly predicted by Interproscan

True negative (TN): element in the not reference pathway and not predicted by Interproscan

False positive (FP): element in the not reference pathway but predicted by Interproscan. Note could also be a valid new prediction

False negative (FN): element in the reference pathway but not predicted by Interproscan.

No EC (noEC): Some elements in the pathways have no EC number record in the database. These cannot be mapped to the interproscan output containing only EC numbers and are left out of the calculations

Sensitivity or true positive rate (TPR): TPR=TP/P=TP/(TP+FN)

Specificity (SPC) or true negative rate: SPC=TN/N=TN/(TN+FP)

Accuracy (ACC) : ACC=(TP+TN)/(TP+FP+FN+TN)

Pathway analyses

Each of the six reference species have a different set of reference pathways. For the following statistical overview, all pathways which are not part of the reference sets are left out: determining positive and negatives is impossible for these.

Species	Culturable	Pathways	Sensitivity	Specificity	Accuracy
CNE	CULT	72	0.63	0.92	0.84
MLR	OBL	75	0.64	0.93	0.85
NCR	CULT	74	0.63	0.92	0.83
PGR	OBL	74	0.63	0.93	0.85
SCE	CULT	71	0.62	0.95	0.86
UMA	OBL	74	0.65	0.92	0.84

7 General discussion

The work presented in this thesis contributes to new high-performance approaches and tools for high-throughput biological and bioinformatics research, mainly using low-cost, standard computer hardware. The main aim is speeding up computation to enable analyses previously unfeasible as a way to approach new biological research questions. The focus has been on genomics, in particular on DNA/RNA sequence analyses (Chapters 2-5) and genome annotation (Chapters 2 and 6).

In addition to new tools, this thesis presents the application of these new tools to the analyses of biological systems (Chapters 5 and 6). In this way different aspects of bioinformatics research are covered: bioinformatics, i.e. the development of new technologies, methods and concepts (Chapters 3 and 6); applied bioinformatics, i.e. the translation of fundamental concepts into applications, packages and/or plugins (Chapters 2 and 4); and computational biology (Chapters 5 and 6), using the tools and/or methods to study biology.

Below, the contributions of the research presented in this thesis are discussed in the context of recent developments and issues in novel computer technologies, notably grid and GPU computing, concerning algorithmic design, software engineering and software testing.

7.1 Computational developments for bioinformatics

One of the major developments in computational infrastructure has been the concept of heterogeneous distributed computing on standard hardware, as effectively used by the still-running SETI@Home project [1–3], i.e. the Search for Extraterrestrial Intelligence (SETI) by analyzing radio signals. SETI demonstrates the viability of the use of a grid of standard desktop computers for scientific data processing. For example, the HTCondor grid management software allows setting up a heterogeneous distributed computing environment on personal computers in local networks (see Chapter 1) [4]. This approach may be advantageous in office surroundings that use many personal computers for relatively modest tasks, such as writing, teaching or administration. The use of office desktop computers for large-scale computations also has its downsides. For one, the hardware is designed for office use and not for large-scale data analyses. As a result, computers, hard drives and other components will fail more often than with normal desktop use. Moreover, desktop computers can overheat, graphics cards (GPUs) can fail, or computers can stop working altogether. All this may happen in desktops much faster than in dedicated servers. Another concern with a grid of desktop computers is that data have to be shared across the network, making it difficult to ensure privacy and security of these data. Despite these limitations, a grid infrastructure can be very beneficial for bioinformatics research. It is a cost-effective way of creating computational power for institutes that otherwise would have less access to such systems. Many of the analyses presented in this thesis (Chapters 2-5) were performed on the

HTCondor grid of the Bioinformatics group at the Institute for Life Science and Technology from the Hanze University of Applied Sciences Groningen. Without the Hanze grid, a large part of the results presented in this thesis would have been much harder, if not impossible, to create in-house.

A second important computational development has been the release of General Purpose Graphics Processing Units (GPGPUs; see Introduction) [5]. A major challenge for the use of GPGPUs in bioinformatics was the application of so-called graphics programming languages, such as OpenGL [6], for tasks they have not been designed for [7]. With the release of the CUDA general purpose programming language and GPUs such as the GeForce 8800 GTX, NVIDIA opened up the platform for a wide array of applications [5]. In bioinformatics, the sequence alignment tool MUMmerGPU [8] was one of the first to demonstrate the benefits of using GPGPU technology for biological data analyses. More recently, the Open Compute language OpenCL has been released, which supports parallel computing on many other platforms, including AMD GPUs, Intel CPUs, ARM chips, etc. For this reason, we created an OpenCL version of PaSWAS and integrated it in Python in the same way as the CUDA version (Chapter 4). Such approaches show the added value of GPGPU computing in bioinformatics. It is likely that many more examples will be accomplished in the future.

7.1.1 Precomputing for predicting pre-miRNAs

MicroRNAs (or miRNAs) are short RNA sequences which regulate gene expression [9,10] and play important roles in biological processes. Computational methods are used to detect these miRNAs and their precursors in genomes [11–13]. In Chapter 2, it is shown that pre-calculation of the minimal free energy (MFE) [14] of tens of millions of sequences brute-force on a large computing grid makes it feasible to screen whole genomes for pre-miRNA candidates. This result demonstrates that distributing calculations on already available infrastructure across many compute units opens up new avenues to do bioinformatics research. The method was applied to the Epstein-Barr genome for the identification of potential pre-miRNA candidates (Chapter 2), and the approach was used by others to identify pre-miRNA candidates in the Tibetan naked carp (Gymnocypris przewalskii) [15]. In silico prediction of (pre-)miRNA candidate sequences can now be supported by additional evidence [16], including Illumina RNA-Seq [17] and CRISPR-Cas9 experimentation [18]. Moreover, better defined sequence characteristics and machine learning approaches will be helpful in further reducing the number of false positives in pre-miRNA candidate prediction.

The underlying idea of pre-calculation can be applied to other computational challenges. In robotics, for example, it is commonly used for scene/map reconstruction and path estimation, because it is computationally impossible to have 100% accuracy in these cases [19]. In

bioinformatics, the number of such existing computational examples may be limited, although the indices built by mapping software [20–22] could be considered pre-calculated databases of sequence information to find related (short) sequences quickly.

7.1.2 Developing GPU applications for bioinformatics research

To get the most out of parallel hardware [23], existing software needs to be rewritten and, in many cases, redesigned. This process can be time-consuming and difficult. The development of PaSWAS, an implementation of Smith-Waterman (SW) sequence alignment on NVIDIA GPUs, demonstrates, however, that such re-development is worth the effort (Chapter 3): the parallel version is much faster than a single-core implementation. The MFold software used to calculate MFEs (Chapter 2) is a valuable yet slow dynamic programming algorithm. It could benefit from a GPU-based implementation [24] and a presumably much faster parallel OpenCL-based extension of MFold could be based on the PaSWAS approach for SW sequence alignment.

GPU-based implementations provide a stable and relatively programmer-friendly platform for parallel processing of data. Moreover, several higher-level programming languages, such as Matlab and R, now offer hardware abstractions for GPUs in the form of software libraries, so that programmers do not need to program GPUs themselves [25–27]. As more of these high-performance compute libraries become available, the need to develop CUDA or OpenCL code decreases, making the compute platforms readily available to a wide range of researchers. In the future, people preparing, for example, R or Matlab code will likely not even be aware that GPUs are used for calculations: a GPU will automatically be used when detected by the software.

The desire for better usability and broader applicability of the CUDA-based application PaSWAS prompted the development of pyPaSWAS (Chapter 4). We integrated the CUDA/OpenCL [28] codebase and combined it with the flexibility of the programming language Python [29,30] to allow it to run on a range of platforms. The code of pyPaSWAS can be used as a library for other applications requiring Smith-Waterman-based alignments. This concept steered the development of Pacasus, a tool requiring high-speed SW to detect erroneous palindromic sequences in long reads with high base-calling error rates after whole-genome amplification (Chapter 5). Pacasus illustrates the attraction of GPGPUs for novel bioinformatics applications.

OpenCL promises to be a generic compute language for parallel systems [31] and should be able to be translated into a single code-base for all parallel devices. However, due to differences in hardware design, the OpenCL code should still be adjusted to get the most out of a particular device [32]. For pyPaSWAS, optimization of the OpenCL code resulted in a significant speed-up when running OpenCL-code optimized for CPUs on a CPU, compared to running the code optimized for the GPU on a CPU. OpenCL optimization is therefore still required to obtain maximum computational efficiency for a specific platform, and code cannot be transferred from

one hardware architecture to another without performance penalties. Writing optimized code also takes time, so a balance between investing time in optimization and running a less-optimized application needs to be found. If future implementations of OpenCL develop into a hardware-independent or hardware-detecting code-base, the use of GPUs for bioinformatics applications is likely to gain further attraction.

7.1.3 Future enhancements

The tools presented in this thesis are not completely finished and probably never will. Feature requests and other issues continue to appear in software [33–35] and the tools presented in this thesis are no exception. Some of the potential additions to and extensions of the tools are discussed below.

PaSWAS has been updated intensively after publication (Chapter 3; https://github.com/swarris/pyPaSWAS/commits/master) during development of pyPaSWAS. A gap extension penalty was added, as well as the OpenCL GPU and CPU implementations. Also, pyPaSWAS has seen updates. After publication of pyPaSWAS (Chapter 4, Release V3.1.1), open source developers from the company StreamComputing have made performance enhancements in the parallel code as well as in the Python code which resulted in Release V3.6. Obviously, there is more to be desired. For example, pyPaSWAS is limited in the length of the sequences it can process by the amount of memory available. Hence it could benefit from distributing computations across multiple GPUs, which would give access to the total memory of all devices used. pyPaSWAS then needs to keep track of which parts of the alignment have been calculated and store these intermediate results in memory or on disk. It is however to be expected that storing these data on disk adds latency because of the required file I/O, which will slow down pyPaSWAS.

Pacasus is developed using pyPaSWAS as a library to detect the chimera introduced in long reads by whole genome amplification (WGA). Two parts of Pacasus are currently not optimized for efficiency, resulting in an implementation which can be improved further. Firstly, pyPaSWAS evaluates all tracebacks through the dynamic programming matrix to return all alignments with the best score, but for Pacasus only the one best hit is needed. As a consequence, the traceback can be calculated during the construction of the matrix, resulting in a speedup of about 30 % because the third phase of PaSWAS can be skipped. Secondly, pyPaSWAS performs an all-versus-all alignment when provided with multiple sequences, whereas Pacasus only needs to align each sequence to itself. Therefore, the current implementation processes a single read at once, which in many cases is not efficient: available memory could allow for detecting palindromes in several sequences in parallel. With short reads <10kb) this inefficiency is large compared to processing long (>10kb) reads. This efficiency of Pacasus can be improved in at least two ways. One adjustment would allow Pacasus to process many (shorter) reads in parallel, requiring fewer

context switches between CPU and GPU. Pacasus also suffers from the O(n2) memory usage, which limits the length of the reads to be aligned to approx. 25,000 bases on GPUs, versus 100,000 bases on CPUs. This limitation on read length is likely to become a serious issue in the near future: both PacBio and Oxford Nanopore technologies are expected to deliver ever-larger read lengths. A banded alignment implementation [36] could help as this will require less memory but might result in false negatives for Pacasus. Whether or not this trade-off between memory usage and loss of precision is a good balance needs to be investigated.

7.2 Software engineering principles

Several software engineering principles (SEPs) [37] have been used in the design, implementation and testing of the new software presented in this thesis without explicit specification. For example, PaSWAS, pyPaSWAS and Pacasus include documentation on how to use the software in README-files and through Wiki pages. pyPaSWAS is provided with test data and a Docker container [38] to install and run the application on any platform. During development of the CUDA and OpenCL code of pyPasWAS, extensive test runs validated the outcomes of the algorithms, verifying that all code-bases produce the same output given the same input and software settings. End-users were also involved in testing the software, with the focus on examples from their respective research topics. Furthermore, the Cytoscape plugin contains several unit-tests and source code for all developed applications has been made available through Open Source licenses. Such SEPs aim to assist in, structure and direct software development to reduce errors, make software more stable, and deliver the required functionality [37,39]. Many SEPs have been developed over the years.

The use of such SEPs in the research setting of this thesis is thought to have had a positive effect on the overall quality (usability, reliability) of the tools developed in this thesis. Although formal scientific demonstration of such effects is notoriously difficult and costly, it is here argued that explicit demonstration and use of SEPs should get more attention and appreciation in bioinformatics. Bioinformatics research and software development requires scientific rigor and peer review. The added value of SEPs for future bioinformatics research and software development is discussed in more detail below.

7.2.1 Development of bioinformatics software

Issues around the testing and validating software, including bioinformatics software, have been around for several years, questioning the quality, stability and predictability [40] of the software involved [41,42]. Small errors in the code have resulted in retractions of high-impact papers [43], even though the results as published raised no suspicion. To prevent such issues with quality, software development in bioinformatics could incorporate SEPs much more formally

than now is often common practice. For example, any application should be divided into parts or modules that can be developed independently and tested separately. These so-called unit tests should be described with well-defined inputs and expected outputs and executed on a variety of relevant data to provide confidence in proper and intended functioning [44]. This approach will create more stable and predictable software, although testing in practice can be challenging [45], time-consuming and costly. The use of other approaches for assessing quality [46], such as coding standards (PEP for Python [47]) and code analytics (Error-Prone [48], IntelliJ [49], Pylint [50]), will further enhance code quality. Any source code should be made publicly available on code sharing sites, such as Github. It is a good development that BioMed Central [51] and Oxford University Press [52] now demand that compiled code is uploaded to open repositories, preferably including the source code. This way, peers can, if so desired, evaluate and test the code. One could even argue that this evaluation should be a mandatory part of the peer-review process. Automated tools to detect code smells [53] are available to detect potential problems and any software submitted or made public should preferably add such analyses upfront. In developing the Cytoscape plugin we started with such analysis on the existing code-base to find and fit problems in the source code before extending the implementation.

7.2.2 Delivery of bioinformatics software to an end-user

Every user of bioinformatics software is helped with a simple, predictable and well-documented interface. Over the years, many graphical user interfaces (GUIs) have been developed that facilitate access to tools in a consistent environment. Some of these GUIs are commercial, such as CLCBio, acquired and further developed by QIAGEN Bioinformatics [54]. Others are open source, such as Galaxy [55]. Power users also benefit from a user-friendly interface. The Canu *de novo* assembler [56] is an excellent example of how experienced users are assisted in the use of a relatively complex command line: Canu auto-detects the type of cluster management software installed, as well as the type(s) of hardware available. Based on the given genome size and other parameters, it distributes calculations automatically. These features make running the software relatively easy and straightforward, also on complex systems as compute clusters. The development of pyPaSWAS (Chapter 4) was motivated by the desire to make PaSWAS (Chapter 3) more user-friendly and better usable by a broader audience. In addition to the additional programming efforts, extensive documentation has been added on how to use the application, including best practices and example uses. Likewise, the Cytoscape plugin (Chapter 6) has several built-in menu options to query the database in an attempt to hide at least some of the complexity of graph databases for biology-trained users. Such approaches for delivering more user-friendly and well-documented software to broaden the applicability and user-base of software developed should be considered part of the research process. They deserve more attention, appreciation and possibly a dedicated platform for publication.

7.2.3 Software engineering skills

The making of good bioinformatics software is a challenge that depends on training and experience. In education, future (and current) bioinformatics software developers should, therefore, be made well aware of existing approaches to and best practices [39,57,58] for the quality, proper testing and appeal to end-users of software.

Recently (in 2018) the Curriculum Task Force of the International Society of Computational Biology (ISCB) Education Committee has proposed to make a clear distinction between a 'bioinformatics engineer' and a 'bioinformatics user' and to adjust educational programs/tracks and training accordingly [59], which I fully support as a bioinformatics researchers and former bioinformatics lecturer.

In the Netherlands, already in 2001 a Bachelor of Applied Sciences (BaSc, in 2017 upgraded to BSc) curriculum for bioinformatics was offered at the Hanze University of Applied Sciences Groningen. This educational track anticipated the need for a bioinformatics engineer as a research technician, in addition to MSc and Ph.D. curricula. In 2002, the Universities of Applied Sciences Arnhem/Nijmegen and Leiden followed. Over the years, bioinformatics software skills have received more attention in biological MSc and Ph.D. curricula, first steered by the Netherlands Bioinformatics Centre (NBIC) and now the Netherlands Bioinformatics and Systems Biology Research school (BioSB) [60]. Internationally, initiatives as GOBLET, the Global Organization for Bioinformatics Learning, Education and Training [61], as well as ELIXIR [62], also gives ample attention to the development of proper bioinformatics software skills as part of scientific training. It is generally acknowledged that there are many challenges in creating and maintaining an appropriate bioinformatics curriculum. There are many topics that compete for limited time. Given the broad nature of knowledge and skills important for bioinformatics, from -omics data (DNA, RNA, protein, metabolites) to databases, statistics and data visualization, decisions on what to give priority are essential. The rapid developments in the field complicate such decisions, as technologies and platforms may quickly become obsolete, sometimes even within the time span of an educational program. It is therefore recommended that focus should not solely go to adjusting current curricula, but also include developing courses, workshops [63] and post-doctoral programs. By adjusting or renewing such elements regularly, bioinformatics practitioners and researchers can stay up to date on current technologies and other developments, and develop a personal education track focused on needs, interests and skill sets. Such life-long learning will be crucial for continued contributions to the field of bioinformatics.

7.3 The road ahead

This thesis shows that the application of new technologies from computer science helps to generate novel approaches for advanced bioinformatics tools and new scientific questions. Given the three major challenges for (applied) bioinformatics in the future described below, the implementation of technologies from computer science is likely to become much more important in the years to come.

7.3.1 Dealing with data volume

The most recent DNA sequencing platform from Illumina, the NovaSeq 6000, currently produces 6 TB of data in two days [64]. By extrapolating the growth of the data generation capacity in biology of the last decade [65], data volumes can safely be predicted to reach the petabyte range on a daily basis in a few years. Compute facilities need to keep up. The use of high-performance infrastructure and technology, such as Hadoop and cloud computing, is becoming common practice [65,66] and tools using hardware such as GPUs are now published frequently [67,68]. Companies as Google submit patents on storing and handling biological data [69,70], indicating commercial interest in these types of data and hence in addressing their capacity problems. In case compute power and storage capacities will not be able to handle the data volumes, researchers are already considering not storing all data and results, but to keep track of the data analytics used, and redo experiments, including sequencing, when needed [71]. Such an approach could work well if the original biological samples are stored properly and the experiments and analyses can be repeated sufficiently fast. This strategy should, however, be considered suboptimal. Appropriate storage of biological material is already a challenge by itself [72]. Moreover, redoing computations carries additional costs that should not be underestimated: repeating a *de novo* assembly of 150,000 hours [73] is not a viable option. The ever-increasing volumes of data will require bioinformatics to adopt appropriate new technology quickly and efficiently. For example, a promising new technology to deal with increasing numbers of genomes is an approach called pan-genomics [74]. In such pan-genomics, graph theory [75,76] is used to store multiple genomes and process these genomes in the database directly [77]. To able to get the most out of graph technology, software for read mappers or variant callers [20,78] will have to be redesigned to work with this new type of storage [79], similar to the need for redesigning software to make efficient use of parallel systems (Chapters 3 and 4). Therefore, the need to redesign software seems a recurrent problem for adopting new technologies in bioinformatics research. As the data volumes require such new technologies, the efforts should be seen as necessary investments in the progress of science, rather than avoidable costs. Moreover, such a redesign could lead to new insights. Possibly computer scientists could focus on design patterns for software development that allow more easy adjustment to any new technology.

7.3.2 Dealing with data types

Not only the data volume poses a significant challenge, but also the number of different data types that are used continues to grow. New data types produced by, for example, automated phenotyping [80–82], sensors based on the IoT [83] or satellite imaging [84] are continuously added to an already wide range of data types available [85]. To be able to address new avenues of research questions, different data types have to be combined. This combination of data types is foremost a technological challenge: each data source has its way of exposing its content (Chapter 6). There are many different standards, not only on how to structure data, such as XML [86] or RDF [87] [88], but also on how machines communicate and exchange data [89]. The latter requires agreement on how (meta)data are annotated. Many approaches to the integration of heterogeneous and multilayer data are in development, such as ontologies or graph databases. An approach in computer science that would seem to deserve more attention for data integration is the use of microservices [90]. The approach of microservices involves making small, independent, applications (services) which perform only single tasks. These services are grouped into a larger application based on a microservice architecture as a counterpart of a single, do-it-all, monolithic architecture [90]. With the help of microservices, a developer can focus on a single task, reducing the complexity of the problem. As a result, the application is easier to maintain and adding new features is a relatively simple task as the feature will be a new microservice. For example, one microservice handles retrieving enzyme information from KEGG [91], and a second microservice stores the results of a read mapping in a database. In theory, each microservice can be built in a different compute language, the one best suited to address the issue at hand. The microservice approach also has downsides. One of the issues is that the microservices need to communicate with each other. The development relies on developers agreeing on how this communication will take place and that they stick to the agreed protocols while continuously developing the microservices. Another issue is that microservices can become entangled in such a way that the overall application becomes unstable and difficult to maintain. Fortunately, several key concepts, such as REST APIs, ontologies and metadata are becoming commonplace. The use of such concepts will facilitate access to data of different types through a generic interface, and should help preventing developers creating yet other data types.

7.3.3 Dealing with complexity

The complexity of data creates the third major challenge that bioinformatics will face in the years to come [92]. Data complexity means that data are heterogeneous, are often obtained from many different sources, can contain missing values and can be contradictory (Chapter 6). The complexity is expected to grow even further, which will make algorithm and tool design increasingly difficult [93]. These three challenges are by themselves complex, but also very much intertwined. To get a grip on these challenges we need to integrate further the research fields

involved, i.e. bioinformatics, applied bioinformatics, computational biology, biology computer science, and applied computer science, as well as different fields of (bio)technology and (bio) engineering. In the post-genomic era, acquiring more data from more organisms in more different ways and in less time is no longer the challenge; analyzing it all is. Contributions to improving the quality of life sustainably will require biology, technology and computer science connected in ways we cannot even imagine yet.

7.4 References

1. Anderson DP, Cobb J, Korpela E, Lebofsky M, Werthimer D. SETI@home: an experiment in public-resource computing. Commun. ACM. 2002;45:56–61.
2. Sullivan III WT, Werthimer D, Bowyer S, Cobb J, Gedye D, Anderson D. A new major SETI project based on Project Serendip data and 100,000 personal computers. IAU Colloq. 161 Astron. Biochem. Orig. Search Life Universe. 1997.
3. Korpela E, Anderson D, Bankay R, Cobb J, Howard A, Lebofsky M, et al. Status of the UC-Berkeley SETI efforts. Conference on Instruments, Methods, and Missions for Astrobiology XIV. 2011.
4. Chapman C, Wilson P, Tannenbaum T, Farrellee M, Livny M, Brodholt J, et al. Condor services for the global grid: interoperability between Condor and OGSA. Proc. 2004 UK e-Science All Hands Meet. Nottingham, UK; 2004. p. 870–7.
5. NVIDIA. NVIDIA Unveils CUDA-The GPU computing revolution begins. 2006. Available from: http://www.nvidia.com/object/IO_37226.html
6. Neider J, Davis T, Woo M. OpenGL programming guide. Addison-Wesley Reading. 1993.
7. Weiguo Liu, Schmidt B, Voss G, Schroder A, Muller-Wittig W. Bio-sequence database scanning on a GPU. Proc. 20th IEEE Int. Parallel Distrib. Process. Symp. IEEE; 2006.
8. Schatz MC, Trapnell C, Delcher AL, Varshney A. High-throughput sequence alignment using Graphics Processing Units. BMC Bioinformatics. 2007.
9. Yang T, Xue L, An L. Functional diversity of miRNA in plants. Plant Sci. 2007.
10. Dragomir MP, Knutsen E, Calin GA. SnapShot: unconventional miRNA functions. Cell. 2018.
11. Doran J, Strauss WM. Bio-informatic trends for the determination of miRNA-target interactions in mammals. DNA Cell Biol. 2007;26.
12. Sarker R, Bandyopadhyay S, Maulik U. An overview of computational approaches for prediction of miRNA genes and their targets. Curr. Bioinform. 2011;6.
13. Tempel S, Tahi F. A fast *ab-initio* method for predicting miRNA precursors in genomes. Nucleic Acids Res. 2012;40:e80.
14. Zuker M, Stiegler P. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Res. 1981;9:133–48.
15. Tong C, Tian F, Zhang C, Zhao K. The microRNA repertoire of Tibetan naked carp *Gymnocypris przewalskii*: A case study in Schizothoracinae fish on the Tibetan Plateau. PLoS One. 2017;12:e0174534.
16. Tian T, Wang J, Zhou X. A review: microRNA detection methods. Org. Biomol. Chem. 2015;13:2226–38.
17. Conesa A, Madrigal P, Tarazona S, Gomez-Cabrero D, Cervera A, McPherson A, et al. A survey of best practices for RNA-seq data analysis. Genome Biol. 2016;17:13.
18. Qiu X-Y, Zhu L-Y, Zhu C-S, Ma J-X, Hou T, Wu X-M, et al. Highly effective and low-cost microRNA detection with CRISPR-Cas9. ACS Synth. Biol. 2018;7:807–13.
19. Whelan T, Kaess M, Johannsson H, Fallon M, Leonard JJ, McDonald J. Real-time large-scale dense RGB-D SLAM with volumetric fusion. Int. J. Rob. Res. 2015;34:598–626.
20. Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv.org. 2013;
21. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat. Methods. 2012;9:357–9.

22. Sović I, Šikić M, Wilm A, Fenlon SN, Chen S, Nagarajan N, et al. Fast and sensitive mapping of nanopore sequencing reads with GraphMap. Nat. Commun. 2016;7:11307.
23. Farber RM. Topical perspective on massive threading and parallelism. J. Mol. Graph. Model. 2011;30:82–9.
24. Januszewski M, Ptok A, Crivelli D, Gardas B. GPU-based acceleration of free energy calculations in solid state physics. Comput. Phys. Commun. 2015;192:220–7.
25. Ishizaki K, Hayashi A, Koblents G, Sarkar V. Compiling and optimizing Java 8 programs for GPU execution. 2015 Int. Conf. Parallel Archit. Compil. IEEE; 2015. p. 419–31.
26. Marowka A. Python accelerators for high-performance computing. J. Supercomput. 2018;74:1449–60.
27. MathWorks. MathWorks GPU Computing [Internet]. Available from: http://nl.mathworks.com/discovery/matlab-gpu.html
28. Munshi A, others. The opencl specification. Khronos OpenCL Work. Gr. 2009;1:l1--15.
29. Python [Internet]. Available from: http://www.python.org
30. Klöckner A, Pinto N, Lee Y, Catanzaro B, Ivanov P, Fasih A. PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. Parallel Comput. 2012;38:157–74.
31. Khronos. OpenCL [Internet]. Available from: www.khronos.org/opencl/
32. Fang J, Varbanescu AL, Sips H. A comprehensive performance comparison of CUDA and OpenCL. 2011 Int. Conf. Parallel Process. IEEE; 2011. p. 216–25.
33. pyPaSWAS issues page [Internet]. Available from: https://github.com/swarris/pyPaSWAS/issues?q=is%3Aissue+is%3Aclosed
34. Pacasus issues page [Internet]. Available from: https://github.com/swarris/Pacasus/issues?q=is%3Aissue+is%3Aclosed
35. Cytoscape Neo4j plugin issues page [Internet]. Available from: https://github.com/corwur/cytoscapeneo4j/issues?q=is%3Aissue+is%3Aclosed
36. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND. Nat. Methods. 2015;12:59–60.
37. Lawlor B, Walsh P. Engineering bioinformatics: building reliability, performance and productivity into bioinformatics software. Bioengineered. 2015;6:193–203.
38. Docker [Internet]. Available from: http://www.docker.com
39. Jiménez RC, Kuzak M, Alhamdoosh M, Barker M, Batut B, Borg M, et al. Four simple recommendations to encourage best practices in research software. F1000Research. 2017;6.
40. Arar ÖF, Ayan K. Deriving thresholds of software metrics to predict faults on open source software: Replicated case studies. Expert Syst. Appl. Pergamon; 2016;61:106–21.
41. Soergel DAW. Rampant software errors may undermine scientific results. F1000Research. 2014;3.
42. Ryan P, Allen A, Teuben P. Schroedinger's code: Source code availability and transparency in astrophysics. Am. Astron. Soc. Meet. Abstr. #231. 2018.
43. Miller G. Scientific publishing. A scientist's nightmare: software problem leads to five retractions. Science. 2006;314:1856–7.
44. Toure F, Badri M, Lamontagne L. Predicting different levels of the unit testing effort of classes using source code metrics: a multiple case study on open-source software. Innov. Syst. Softw. Eng. 2018;14:15–46.
45. Prado MP, Vincenzi AMR. Towards cognitive support for unit testing: A qualitative study with practitioners. J. Syst. Softw. 2018;141:66–84.
46. Mukherjee S. Source Code Analytics With Roslyn and JavaScript Data Visualization. Berkeley, CA; 2016. p. 15–44.
47. Python.org. PEP [Internet]. Available from: https://www.python.org/dev/peps/
48. Google. Error-Prone [Internet]. Available from: https://errorprone.info/
49. JetBrains. IntelliJ [Internet]. Available from: https://www.jetbrains.com/idea/
50. Pylint [Internet]. Available from: https://www.pylint.org/
51. BMC. BMC Bioinformatics: Instruction to authors [Internet]. Available from: https://bmcbioinformatics.biomedcentral.com/submission-guidelines/preparing-your-manuscript/software-article
52. Bioinformatics. Bioinformatics: Instruction to authors [Internet]. Available from: https://academic.oup.com/bioinformatics/pages/instructions_for_authors

53. Mansoor U, Kessentini M, Maxim BR, Deb K. Multi-objective code-smells detection using good and bad design examples. Softw. Qual. J. 2017;25:529–52.

54. CLCBio Company. [Internet]. Available from: htttp://www.clcbio.com.

55. Goecks J, Nekrutenko A, Taylor J. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. 2010;11:R86.

56. Koren S, Walenz BP, Berlin K, Miller JR, Bergman NH, Phillippy AM. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Genome Res. 2017;27:722–36.

57. Wilson G, Aruliah DA, Brown CT, Chue Hong NP, Davis M, Guy RT, et al. Best practices for scientific computing. PLoS Biol. 2014;12:e1001745.

58. Leprevost F da V, Barbosa VC, Francisco EL, Perez-Riverol Y, Carvalho PC. On best practices in the development of bioinformatics software. Front. Genet. 2014;5:199.

59. Mulder N, Schwartz R, Brazas MD, Brooksbank C, Gaeta B, Morgan SL, et al. The development and application of bioinformatics core competencies to improve bioinformatics training and education. PLoS Comput. Biol. 2018;14:e1005772.

60. BioSB [Internet]. Available from: https://www.biosb.nl/

61. Atwood TK, Bongcam-Rudloff E, Brazas ME, Corpas M, Gaudet P, Lewitter F, et al. GOBLET: The Global Organisation for Bioinformatics Learning, Education and Training. Welch L, editor. PLOS Comput. Biol. Public Library of Science; 2015;11:e1004143.

62. Elixir [Internet]. Available from: https://www.elixir-europe.org/

63. Software Carpentry [Internet]. Available from: https://software-carpentry.org/

64. Illumina. NovaSeq 6000 Sequencing System [Internet]. Available from: https://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/novaseq-6000-system-specification-sheet-770-2016-025.pdf

65. Luo J, Wu M, Gopukumar D, Zhao Y. Big Data application in biomedical research and health care: a literature review. Biomed. Inform. Insights. 2016;8:1.

66. Ko G, Kim P-G, Yoon J, Han G, Park S-J, Song W, et al. Closha: bioinformatics workflow system for the analysis of massive sequencing data. BMC Bioinformatics. 2018;19:43.

67. García-Calvo R, Guisado J, Diaz-del-Rio F, Córdoba A, Jiménez-Morales F. Graphics Processing Unit–enhanced genetic algorithms for solving the temporal dynamics of gene regulatory networks. Evol. Bioinforma. 2018;14:11.

68. Kovac T, Haber T, Reeth F Van, Hens N. Heterogeneous computing for epidemiological model fitting and simulation. BMC Bioinformatics. 2018;19:101.

69. Google. Compressing, storing and searching sequence data. 2017. Patent US20170323052A1.

70. Sqream Technologies Ltd. Method and system for compressing genome sequences using graphic processing units. 2016. Patent US20180011870A1

71. Hart EM, Barmby P, LeBauer D, Michonneau F, Mount S, Mulrooney P, et al. Ten simple rules for digital data storage. Markel S, editor. PLOS Comput. Biol. 2016;12:e1005097.

72. Huang L-H, Lin P-H, Tsai K-W, Wang L-J, Huang Y-H, Kuo H-C, et al. The effects of storage temperature and duration of blood samples on DNA and RNA qualities. PLoS One. 2017;12:e0184692.

73. Nowoshilow S, Schloissnig S, Fei J-F, Dahl A, Pang AWC, Pippel M, et al. The axolotl genome and the evolution of key tissue formation regulators. Nature. 2018;554:50–5.

74. Marschall T, Marz M, Abeel T, Dijkstra L, Dutilh BE, Ghaffaari A, et al. Computational pan-genomics: status, promises and challenges. Brief. Bioinform. 2016;19:bbw089.

75. Biggs N, Lloyd EK, Wilson RJ. Graph Theory, 1736-1936. Oxford University Press; 1976.

76. Deo N. Graph theory with applications to engineering and computer science. Courier Dover Publications; 2017.

77. Sheikhizadeh S, Schranz ME, Akdel M, de Ridder D, Smit S. PanTools: representation, storage and exploration of pan-genomic data. Bioinformatics. 2016;32:i487–93.

78. Valenzuela D, Norri T, Välimäki N, Pitkänen E, Mäkinen V. Towards pan-genome read alignment to improve variation calling. BMC Genomics. 2018;19:87.

79. Zekic T, Holley G, Stoye J. Pan-Genome Storage and Analysis Techniques. Humana Press. 2018. p. 29–53.

80. Czedik-Eysenberg A, Seitner S, Güldener U, Koemeda S, Jez J, Colombini M, et al. The 'PhenoBox', a flexible, automated, open-source plant phenotyping solution. New Phytol. 2018;

81. Lee U, Chang S, Putra GA, Kim H, Kim DH. An automated, high-throughput plant phenotyping system using machine learning-based plant segmentation and image analysis. PLoS One. 2018;13:e0196615.

82. Krajewski P, Chen D, Ćwiek H, van Dijk ADJ, Fiorani F, Kersey P, et al. Towards recommendations for metadata and data handling in plant phenotyping. J. Exp. Bot. 2015;66:5417–27.

83. Oppitz M, Tomsu P. Internet of Things. Invent. Cloud Century. Cham. 2018. p. 435–69.

84. Rudd JD, Roberson GT, Classen JJ. Application of satellite, unmanned aircraft system, and ground-based sensor data for precision agriculture: a review. 2017 ASABE Annu. Int. Meet. 2017.

85. Rigden DJ, Fernández XM. The 2018 Nucleic Acids Research database issue and the online molecular biology database collection. Nucleic Acids Res. 2018;46:D1–7.

86. W3C. XML specifications [Internet]. Available from: https://www.w3.org/TR/REC-xml/

87. W3C. RDF specifications [Internet]. Available from: https://www.w3.org/RDF/

88. Koehorst JJ, van Dam JCJ, Saccenti E, Martins dos Santos VAP, Suarez-Diez M, Schaap PJ. SAPP: functional genome annotation and analysis through a semantic framework using FAIR principles. Hancock J, editor. Bioinformatics. Oxford University Press; 2018;34:1401–3.

89. API technical and data standards [Internet]. Available from: https://www.gov.uk/guidance/gds-api-technical-and-data-standards

90. Dragoni N, Giallorenzo S, Lafuente AL, Mazzara M, Montesi F, Mustafin R, et al. Microservices: yesterday, today, and tomorrow. Present Ulterior Softw. Eng. Cham; 2017. p. 195–216.

91. Kanehisa M, Goto S. KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Res. 2000;28:27–30.

92. Fan J, Han F, Liu H. Challenges of Big Data analysis. Natl. Sci. Rev. 2014;1:293–314.

93. Sivarajah U, Kamal MM, Irani Z, Weerakkody V. Critical analysis of Big Data challenges and analytical methods. J. Bus. Res. 2017;70:263–86.

Summary

Bioinformatics and computational biology are driven by growing volumes of data in biological systems that also tend to increase in complexity. The research presented in this thesis focuses on the need to analyze such data volumes in such complexity. The results show that the application of high-performance compute technologies, preferably combined with low-cost hardware, is a successful approach to generate new bioinformatics approaches that allow addressing new types of data analyses and research questions in biology.

An overview of the technologies and recent developments in biology and computer science relevant for this thesis (Chapter 1) identifies current high-throughput sequencing platforms as a key technology. Sequencing platforms now deliver data sets up to terabytes in size for elucidating genome structure, gene content, gene activity, as well as gene variants. The concepts and technologies from computer science to handle these large amounts of data include (a) grid technologies for compute parallelization while making more efficient use of existing low-cost infrastructure; (b) graphics cards for increased compute power and (c) graph databases for large data volume storage and advanced methods for analyses. This thesis presents novel applications and added value of these three concepts for bioinformatics research.

Small RNAs are important regulators of genome function, yet their prediction in genomes is still a major computational challenge (Chapter 2). They tend to have a minimal free energy (MFE) significantly lower than the MFE of non-small RNA sequences with the same nucleotide composition. Evaluation of many MFEs is, however, too compute-intensive for genome-wide screening. With a local grid infrastructure of desktop computers, MFE distributions of a very large collection of sequence compositions were pre-calculated and used to determine the MFE distribution for any given sequence composition by interpolation. This approach allows on-the-fly calculation for any candidate sequence composition and makes genome-wide screening with this characteristic of a pre-miRNA sequence feasible. This way, MFE evaluation can be added as a new parameter for genome-wide selection of potential small RNA candidates (Chapter 2). The concept of large-scale pre-calculation of compute-intensive parameters is one of the options for future bioinformatics analyses.

Sequence alignment is essential in the analysis of next-generation sequencing data. The gold standard for sequence alignment is the Smith-Waterman (SW) algorithm. Existing implementations of the full SW algorithm are either not fast enough, or limited to dedicated tasks, usually to optimize for speed, whereas popular heuristic SW versions (such as BLAST) suffer from statistical issues. Graphics hardware is well-suited to speed up SW alignments, but SW on

graphics cards does not report the alignment details desired by biologists for further analysis. This thesis presents the CUDA-based Parallel SW Alignment Software (PaSWAS) (Chapter 3). PaSWAS gives (a) easy access to the computational power of NVIDIA-based graphics cards for high-speed sequence alignments, (b) information such as score, number of gaps and mismatches with the accuracy of the full SW algorithm and (c) a report of multiple hits per alignment. Two use cases show the usability and versatility of the new parallel Smith-Waterman implementation for bioinformatics analyses. It demonstrates the added value of the use of low-cost graphics cards in bioinformatics software.

To further promote the use of PaSWAS, a new implementation, pyPaSWAS, provides the SW sequence alignment code fully packed in Python and the more widely accepted OpenCL language (Chapter 4). Moreover, pyPaSWAS now supports an affine gap penalty. This way, pyPaSWAS presents an easy Python-based environment for accurate and retrievable parallel SW sequence alignments on GPUs and multi-core systems. The strategy of integrating Python with high-performance parallel compute languages to create a developer- and user-friendly environment is worth to be considered for other computationally-intensive bioinformatics algorithms.

Thanks to the accuracy and retrieval characteristics of (py)PaSWAS, it was noted that long sequencing reads on the PacBio platform can contain many artificial palindromic sequences. These palindromes are due to errors introduced by whole-genome amplification (WGA). Next-generation sequencing requires sufficient amounts of DNA. If not available, WGA is routinely used to generate the amounts of DNA required. The introduction of artificial palindromic sequences hampers assembly and severely limits the value of long sequencing reads. Pacasus is a novel software tool to identify and resolve such artificial palindromic sequences in long sequencing reads (Chapter 5). Two use cases show that Pacasus markedly improves read mapping and assembly of WGA DNA. In comparison, the quality of mapping and assembly is similar to the quality obtained with non-amplified DNA. Therefore, with Pacasus, long-read technology becomes feasible for the sequencing of samples for which only very small amounts of DNA are available, such as single cells or single chromosomes.

Numerous tools and databases exist to annotate and investigate the functions encoded in properly assembled genomes, such as InterProScan, KEGG, GO and many more. Comparisons of functionalities across multiple genomes is, however, not trivial. The concept of graph databases is a promising novel approach from computer science for such multi-genome comparisons. For a data set of all (> 150,000) genes of 17 fungal species functionally annotated with InterProScan, the associated KEGG, GO and annotation data are imported and interconnected in a new Neo4j graph database (Chapter 6). Relationships in this database are visualized and mined with a newly refurbished and extended Neo4j plugin for Cytoscape. Inspection of (sub)graphs of functional annotations is an attractive way to compare and group functional annotation across species. In the

use case of the seventeen fungal genomes, it helped to outline, compare and explain details of the life style of groups of individual species.

The general discussion of this thesis provides an outlook on the future of bioinformatics in the context of the results here presented (Chapter 7). A grid infrastructure is recommended as a feasible and attractive cost-effective strategy to create compute power, as is the further inclusion of graphics cards. Full implementation of graph technology is considered necessary for advancing bioinformatics. The work presented in this thesis also shows that use of grids, graphics cards and graph technology imply the redesign of existing software applications. To be able to create novel stable, predictable and user-friendly applications in bioinformatics, formal training in software engineering principles is highly recommended. Courses and other programs are necessary for the life-long learning that will be crucial for the future of bioinformatics. The main challenges for bioinformatics in the years to come are all data centered: issues with growing data volumes, with more data types and with higher data complexity. To deal with these challenges, further integration of now separate fields of science is warranted in ways we cannot even image yet.

Samenvatting

Bioinformatica en computationele biologie worden geconfronteerd met voortdurend groeiende hoeveelheden data die ook steeds complexer worden. Het onderzoek in dit proefschrift richt zich op de wens en noodzaak om deze grote en complexe datasets goed te kunnen analyseren. De resultaten laten zien dat het gebruik van hoogwaardige rekentechnieken op -bij voorkeur- goedkope hardware kan leiden tot nieuwe aanpakken voor data-analyse in de bioinformatica en tot nieuwe onderzoeksvragen in de biologie.

Hoofdstuk 1 geeft een overzicht van recente ontwikkelingen in de biologie en de informatica met betrekking tot technologieën die relevant zijn voor het onderzoek in dit proefschrift, waarbij de ontwikkeling van de high-throughput sequencing platformen als belangrijkste wordt aangemerkt. Deze platformen genereren tot terabytes aan data welke inzicht geven in (structuren in) genomen, genen, gen-activiteiten alsmede genetische variatie. Technologieën uit de informatica die het mede mogelijk maken om deze datasets goed te verwerken zijn (a) grid infrastructuren voor de efficiënte parallelle verwerking op bestaande goedkope infrastructuur (b) grafische kaarten voor meer rekenkracht en (c) graph databases voor grootschalige opslag en geavanceerde analysetechnieken. Dit proefschrift presenteert nieuwe toepassingen op basis van deze drie concepten en hun toegevoegde waarde voor de bioinformatica.

Kleine RNA-moleculen zijn belangrijk voor het reguleren van vele functies van het genoom, maar het voorspellen van deze kleine moleculen in het genoom is nog steeds een computationele uitdaging (Hoofdstuk 2). Deze RNA-moleculen hebben in de meeste gevallen een significant lagere minimale vrije energie (minimal free energy, MFE) vergeleken met andere RNA-moleculen met dezelfde nucleotidesamenstelling. De evaluatie van een groot aantal MFE-waarden maakt een scan van een heel genoom te rekenintensief. Door gebruik te maken van een lokale grid infrastructuur gebaseerd op een netwerk van desktopcomputers is een grote set van MFE-waarden uitgerekend, waarmee de MFE-verdeling bepaald kan worden voor elke gewenste nucleotidesequentie met behulp van interpolatie. Hiermee is het mogelijk om voor elke kandidaatsequentie in korte tijd de MFE-verdeling te bepalen, waardoor het analyseren van een heel genoom haalbaar wordt. De MFE-waarde kan toegevoegd worden als nieuwe parameter voor het classificeren van RNA-moleculen (Hoofdstuk 2). Het concept van het vooraf grootschalig uitrekenen van rekenintensieve berekeningen is een mogelijkheid voor toekomstig onderzoek aan grote datasets, bijvoorbeeld in de bioinformatica.

Het vergelijken van sequenties is een essentieel onderdeel bij het analyseren van sequentiedata. De gouden standaard hiervoor is het Smith-Waterman (SW) algoritme. Bestaande implementaties

van het volledige SW-algoritme zijn niet snel genoeg, of gelimiteerd tot een bepaalde taak die meestal geoptimaliseerd is voor snelheid. Andere aanpakken, zoals BLAST, zijn heuristisch en hebben ingebouwde statistische onbetrouwbaarheden. Grafische kaarten zijn heel geschikt voor SW analyses, maar bestaande implementaties geven niet voldoende details voor verdere biologische analyses of interpretaties. Dit proefschrift presenteert (Hoofdstuk 3) een op de CUDA programmeertaal gebaseerde parallelle implementatie van het SW algoritme: Parallele Smith-Waterman Alignment Software (PaSWAS). PaSWAS geeft (a) makkelijk toegang tot de rekenkracht van op NVIDIA-technologie gebaseerde grafische kaarten voor snelle analyses, (b) informatie over bv. de score, het aantal gaten en het aantal niet-overeenkomstige letters met behoud van de volledige accuratesse van het SW-algoritme en (c) de mogelijkheid om meerdere resultaten per vergelijking te geven. Twee voorbeelden laten zien dat PaSWAS een bruikbare en veelzijdige nieuwe parallelle implementatie van het SW-algoritme is voor analyses in de bioinformatica. Het laat ook zien dat het gebruik van relatief goedkope grafische kaarten meerwaarde heeft voor de bioinformatica.

Om het gebruik van PaSWAS verder te stimuleren/vergemakkelijken heeft een nieuwe implementatie, pyPaSWAS, de SW analyses samengevoegd met Python code. Deze implementatie bevat ook een versie in de breed geaccepteerde OpenCL programmeertaal (Hoofdstuk 4). Ook heeft pyPaSWAS de mogelijkheid om gebruik te maken van de meest geavanceerde wijze van het meenemen van gaten in de vergelijking (‘affine gap penalty’). pyPaSWAS geeft op deze wijze via een toegankelijke Python omgeving de mogelijkheid om accurate en volledige SW analyses uit te voeren op verschillende merken grafische kaarten en computers met meerdere CPU kernen. De strategie van het samenvoegen van Python met parallelle rekentalen om tot een ontwikkelaar(s)- en gebruikersvriendelijke omgeving te komen kan overwogen worden voor toekomstige rekenintensieve taken in de bioinformatica.

Dankzij de accuratesse en informatie van (py)PaSWAS werd zichtbaar dat lange sequenties afkomstig van het PacBio sequentieplatform op grote schaal palindromen kunnen bevatten. Deze palindromen bleken kunstmatig tot stand te komen tijdens het proces van genoomvermeerdering (whole genome amplication (WGA)). De huidige sequentieplatformen vereisen voldoende DNA om de sequentie te kunnen bepalen. Als de benodigde hoeveelheid DNA niet aanwezig is, wordt routinematig WGA ingezet om meer DNA te genereren. Door de introductie van kunstmatige palindromen kunnen dergelijke lange sequenties niet goed ingezet worden bij de opheldering van genoomstructuren (‘assembly’). Dit beperkt de inzet van de WGA-technologie enorm. Pacasus is nieuwe software die het mogelijk maakt om deze palindromen te detecteren en hun informatie ook te gebruiken (Hoofdstuk 5). Voorbeelden laten zien dat Pacasus belangrijke verbeteringen oplevert bij de kartering (‘mapping’) van WGA DNA-sequenties op een genoom en bij het de novo assembleren van een genoom. De resultaten zijn goed vergelijkbaar met de resultaten

van DNA dat niet via WGA is verkregen. Met Pacasus wordt het dus mogelijk om lange DNA-sequenties te gebruiken van biologische specimen waarvan maar heel weinig DNA beschikbaar is, zoals in het geval van een enkele cel of een enkel chromosoom.

Er zijn talloze softwarepakketten en databases beschikbaar, zoals InterProScan, KEGG en GO, om de functies te analyseren in een genoom waarvan de structuur is opgehelderd. Het vergelijken van functies tussen verschillende genomen is echter niet triviaal. Het concept van graph databases is een nieuwe en interessante aanpak voor dit soort meervoudige-genoomvergelijkingen. Voor een dataset van alle (> 150.000) genen van zeventien schimmels, die functioneel beschreven (geannoteerd) zijn met InterProScan, zijn alle KEGG, GO en andere functionele beschrijvingen in een Neo4j graph database gecombineerd (Hoofdstuk 6). Relaties in deze database zijn gevisualiseerd en doorzocht via een aangepaste en uitgebreide Neo4j plug-in voor Cytoscape. Analyse van (sub)graphs van functionele annotaties is een attractieve methode voor het vergelijken en groeperen van functionele annotaties van meerdere soorten. In het voorbeeld van zeventien schimmelgenomen werd het mogelijk om nieuwe vergelijkingen te maken tussen groepen van verschillende organismen en nieuwe inzichten te verkrijgen over de verschillende levensstijlen van die groepen van organismen.

De algemene discussie van dit proefschrift (Hoofdstuk 7) geeft vergezichten op de toekomst van bioinformatica in het licht van de hier gepresenteerde resultaten. Een grid infrastructuur is een aantrekkelijke en goedkope wijze om de beschikking te krijgen over meer rekenkracht. De toevoeging van grafische kaarten aan dit geheel vergroot de rekenkracht verder. Voor veel vraagstukken in de bioinformatica is het even aantrekkelijk als noodzakelijk om meer gebruik te maken van graph-technologieën. Het proefschrift laat zien dat voor het gebruik van grids, grafische kaarten en graph technologieën het nodig is om bestaande softwareapplicaties te herschrijven. Om de vereiste nieuwe, stabiele, voorspelbare en gebruikersvriendelijke bioinformatica-applicaties te maken is training in software engineering principes zeer aan te raden. Cursussen en andere lesprogramma's zijn nodig voor het leven-lang-leren dat cruciaal zal zijn voor de toekomst van de bioinformatica. Deze toekomst draait rond data: uitdagingen met de verdere groei van datasets met meer verschillende soorten data en toenemende complexiteit bepalen het beeld. Om met deze uitdagingen om te kunnen gaan zullen nu nog gescheiden onderzoeksgebieden geïntegreerd moeten worden op manieren die we ons nu nog niet kunnen voorstellen.

Acknowledgements

This thesis is the result of a long journey, during which many people have contributed to, helped with, or have influenced me and my work. Or may have been influenced by me and my work. I here try to acknowledge as many people as possible, but in case I forget any, I offer my sincere apologies in advance.

First and foremost, I like to thank Jan Peter Nap for coaching me into a PhD project and, more importantly, for his supervision, patience and guidance. We might not always have agreed on everything, but the trust you gave me in finding my way and letting me structure this PhD with topics I found appealing has been of great importance. You also taught me how to become a better researcher. And indeed, scientific writing will never be one of my favourite activities. The support from the management team of the Hanze Institute for Life Science & Technology, including Ida, Rob and Victorine, has been key in providing me the opportunity to work on my research projects and include these in my classes. It made the lectures and lab work more attractive for the students and helped my research.

I thank all my students, notably Jerven, Heleen, Iris, Wil, Patrick and Marc Jan, for all intense discussions on GPU technology, dingoes and machine learning. Several students are a co-author on papers in this thesis, and I am grateful that Sander, Feyruz, Iwe and Marcel helped me with these topics. My direct colleagues at Hanze Applied Bioinformatics have been essential for the work presented: Michiel never stopped asking questions and was always willing to brainstorm and help out. Piet, your help has been invaluable. You set up the grid infrastructure for me on one of the best computer networks I have ever seen. You never got angry for freezing up a computer or making the temperature in the lecture rooms rise beyond tropical values. For questions related to statistics, Wim was always happy to help. Arne made highly appreciated contributions in code and comments. I have really enjoyed my time at the Institute for Life Science and Technology. I have had good fun with colleagues and students from the other three educational programs. Suzanne, Josina, Grietinus, Peter, Henk, Wietske and the students Julia and Tessa all deserve special mention.

I started my PhD project with Willem Stiekema as intended promotor. As my PhD took a bit longer than planned, he left science before the finalisation of my thesis. Dick de Ridder as professor of Bioinformatics at WUR was willing to take over the role as promotor. I am very happy Dick accepted this PhD project: your technological knowledge and insights are excellent and have been instrumental for finishing. You have not only commented on my writing, but you have also been willing to clone my repository and perform extensive user-tests on the software. Your push-to-finish mentality combined with your friendly personality helped a great deal in getting this thesis job done!

During my PhD I have had the opportunity to visit the University of New South Wales twice for a couple of months. I enjoyed my time at the lab there, and I am grateful Bruna, Bill and Alan gave me a chance to work in the most important lab on dingo research in the world. I also look back with joy on our walks in the Blue Mountains and salmon on the barbie. Also, Kylie, Katherina and the other team members of BABS taught me a lot about working in the lab and the challenges you face when performing biological research.

After almost ten years working at the Hanze UAS, I switched to the job of bioinformatics researcher at Applied Bioinformatics (WUR). I like to thank Gabino for hiring me and allowing me to integrate my earlier PhD research with my new work assignments. Otherwise I would have had much more struggle to finish this PhD project. Within these projects I worked closely with Elio and Henri on Pacasus. The support I also got from the other team members, Linda, Jan, Thamara, Bas, Aalt-Jan, Saulo, Ronald and Sander helped me to stay focused and enjoy my PhD. Also I have appreciated the help and support I received from the chair group staff, mainly Harm and Sandra. I will never forget my office roommate and fellow-PhD-enthusiast: Sevgin. Although you seemed to think I was doing a PhD 'just as a hobby', I will treasure the talks we had on all things concerning doing a PhD and beyond.

For one project at WUR, Theo asked me to help a PhD student with some bioinformatics work. This work got a little bit out of hand and resulted in a comprehensive genome comparison of several fungal species. Bart, it was an honour and a pleasure to work with you!

My PhD could not have been possible with the help of many other scientists. In particular I like to thank Ana and Roshan for their important contributions to PaSWAS and pyPaSWAS. The same gratitude goes to Paul and Kateryna for their work on the gorilla Y chromosome.

Sadly, we lost two Hanze colleagues, Ko and Jos, during my time at Hanze. With the passing of Alan, I lost a friend, and the dingo world lost one of its key supporters.

As with any PhD, but especially with one that took place for a large part in spare time, people in my personal life played important roles. My dearest friends Michael and Ivor were there from the start and supported me all the way. Coming from IT backgrounds, they also helped in shaping the computer science part of my PhD. My family, Arja, Daphne and my parents, were very supportive and helpful where they could. Halfway through my PhD I met Judith and her loving family. Anny and Harry helped out by taking care of Tobias when needed, and Arthur was so kind to allow me to work in his apartment. But most of all I am forever grateful for the love and support I got from my partner Judith. It was not always easy, and she had to make sacrifices so I could work on my PhD and finish my thesis. Our sons Tobias and Kasper have now no idea, but hopefully someday they will read this thesis to see what their dad was up to in their first years.

Curriculum vitae

Sven Warris was born in Assen, The Netherlands on December 10th, 1975. His family moved to Emmen in 1976, where he had his primairy education and he started pre-university (VWO) education. Halfway, the family moved to Uden, where he graduated in 1994. The same year, Sven started the study Computer Science at the University of Groningen, as the prerequisite for the MSc (Drs) program Technical Cognitive Science, currently the MSc program Artificial Intelligence. Sven graduated in 2002 at the Neurobiophysics department of the University of Groningen with a thesis on biologically plausible neural networks. He worked as software engineer before joining Hanze University of Applied Sciences Groningen in 2004 as lecturer/researcher, investigating topics in applied bioinformatics and teaching computer science and data science-related topics to bachelor (now BSc) students of (applied) bioinformatics. In 2007, Sven started the PhD project the results of which are presented in this thesis, in addition to ongoing teaching responsibilities. As part of his PhD, he spent in total four-and-a-half months (2008 and 2010) at the University of New South Wales, Sydney, Australia. In 2012, he started as researcher bioinformatics in the Cluster Applied Bioinformatics of Wageningen University & Research, focusing on applying computer/data science technologies to (plant-)biological questions and research topics, part of which also features in this thesis. He will continue his work as a senior researcher in the same cluster.

List of publications

Warris, Sven; Boymans, Sander; Muiser, Iwe; Noback, Michiel; Krijnen, Wim; Nap, Jan-Peter; ***Fast selection of miRNA candidates based on large-scale pre-computed MFE sets of randomized sequences***, BMC Research Notes. 2014 (Chapter 2)

Warris, Sven; Yalcin, Feyruz; Jackson, Katherine JL; Nap, Jan Peter; ***Flexible, fast and accurate sequence alignment profiling on GPGPU with PaSWAS***, PloS ONE. 2015 (Chapter 3)

Vanheule, Adriaan; Audenaert, Kris; **Warris, Sven**; van de Geest, Henri; Schijlen, Elio; Höfte, Monica; De Saeger, Sarah; Haesaert, Geert; Waalwijk, Cees; van der Lee, Theo; ***Living apart together: crosstalk between the core and supernumerary genomes in a fungal plant pathogen***, BMC Genomics. 2016

van der Wolf, JM; Kastelein, P; Krijger, MC; Hendriks, MJA; van der Lee, TAJ; Taparia, T; **Warris, S**; Stol, W; Amsing, J; ***Suppressiveness of casing material against bacterial blotch***, KNPV meeting Fytobacteriologie. 2016

van der Wolf, JM; Kastelein, P; Krijger, MC; Hendriks, MJA; Baars, JJP; Amsing, JGM; van der Lee, TAJ; **Warris, S**; ***Characterization of Pseudomonas species causing brown blotch of* Agaricus bisporis.**, 19th ISMS Congress proceeding. 2016

Waalwijk, Cees; Vanheule, Adriaan; Audenaert, Kris; Zhang, Hao; **Warris, Sven**; van de Geest, Henri; van der Lee, Theo; ***Fusarium in the age of genomics***, Tropical Plant Pathology. 2017

van der Wolf, JM; Coipan, EC; Kastelein, P; Krijger, MC; Tom, Jolanda; Riksen, Natasja; Nijhuis, EH; **Warris, S**; Fokkema, Jenny; ***Suppressiveness of Potato against* Dickeya solani**, Euphresco III Dickeya/Pectobacterium workshop. 2017

Schulze, Stefan; Urzica, Eugen; Reijnders, Maarten JMF; Geest, Henri; **Warris, Sven**; Bakker, Linda V; Fufezan, Christian; Martins dos Santos, Vitor AP; Schaap, Peter J; Peters, Sander A; ***Identification of methylated GnTI-dependent N-glycans in* Botryococcus brauni**, New Phytologist, 215, 4, 1361-1369, 2017

Warris, Sven; Timal, N Roshan N; Kempenaar, Marcel; Poortinga, Arne M; van de Geest, Henri; Varbanescu, Ana L; Nap, Jan-Peter; ***pyPaSWAS: Python-based multi-core CPU and GPU sequence alignment***, PloS ONE. 2018 (Chapter 4)

Warris, Sven; Dijkxhoorn, Steven; van Sloten, Teije; van de Vossenberg, Bart TLH; ***Mining functional annotations across species***, bioRxiv. 2018 (Chapter 6)

Cochran, Alyssa T; Bauer, Jemma; Metcalf, Jessica L; Lovecka, Petra; Sura de Jong, Martina; **Warris, Sven**; Mooijman, Paul JW; Van der Meer, Ingrid; Knight, Rob; Pilon-Smits, Elizabeth

AH; ***Plant Selenium Hyperaccumulation Affects Rhizosphere: Enhanced Species Richness and Altered Species Composition***, Phytobiomes. 2018

Warris, Sven; Schijlen, Elio; van de Geest, Henri; Vegesna, Rahulsimham; Hesselink, Thamara; te Lintel Hekkert, Bas; Perez, Gabino Sanchez; Medvedev, Paul; Makova, Kateryna D; de Ridder, Dick; C***orrecting palindromes in long reads after whole-genome amplification***, BMC Genomics. 2018 (Chapter 5)

Warris, Sven; van der Lee, T; ***Professional, easy to use and robust bioinformatic tools***, Scientific Symposium FAIR Data Sciences for Green Life Sciences. 2018

B.T.L.H. van de Vossenberg & **S. Warris**, H.D.T. Nguyen, M. van Gent-Pelzer, D.L. Joly, H.C. van de Geest, P.J.M. Bonants, D.S. Smith, C.A. Lévesque, T.A.J. van der Lee, ***Comparative genomics of chytrid fungi reveal insights into the obligate biotrophic and pathogenic lifestyle of*** **Synchytrium endobioticum**, Nature Scientific Reports. 2019 (Basis for Chapter 6)

Propositions

1. General-Purpose Graphics Processing Unit (GPGPU) technologies are highly undervalued in bioinformatics research.
   (this thesis)

2. Proven applicability and novelty are the chicken and egg of bioinformatics.
   (this thesis)

3. More funds should be moved from traditional research-oriented universities to Universities of Applied Sciences for higher return-on-investment and higher impact of research on society.

4. Human intelligence as basis for Artificial Intelligence (AI) is too scary to be considered safe.

5. Given the current state of biological databases, FAIR stands for Freakishly Ambiguous and Incomplete Resources.

6. Dreaming of a next Frisian Eleven Cities Tour is the ultimate denial of climate change.

Propositions belonging to the thesis, entitled

Application of high performance compute technology in bioinformatics

Sven Warris

Wageningen, 22 October 2019

The research described in this thesis was financially supported by the Hanze University of Applied Sciences Groningen. Sections of the work have been part of the research programme “Application of High-Performance, Low-Cost Biocomputing in Genomics (BioCOMP)”, which was a SIA RAAK-PRO project financed in part by the Task Force for Applied Research SIA, now a unit of the Dutch Research Council (NWO).

Additional financial support for printing this thesis from the Research Centre BioBased Economy of the Hanze University of Applied Sciences Groningen is gratefully acknowledged.