Using SIMD to accelerate DNA Design

RAJ BAPAT, University of California, Davis
DAVID DOTY, University of California, Davis

DNA sequence design stands as a cornerstone in advancing scientific endeavors across a multitude of domains, including medicine, biotechnology, and synthetic biology. Through meticulous engineering of DNA sequences, researchers can realize breakthroughs such as novel therapeutics, sustainable biomaterials, and precisely tailored organisms. However, the complexity of DNA design demands robust and high performance computational tools to efficiently navigate this vast landscape of possibilities and speed up research and development.

Within this context, the NUAD (NUcleic Acid Designer) project empowers researchers to impose intricate constraints on DNA or RNA nanostructures crafted from synthetic genetic material. Unlike standalone programs like NUPACK, NUAD prioritizes expressiveness over simplicity, enabling power users to define nuanced requirements such as specific energy thresholds and binding domain constraints. NUAD enables the creation of bespoke DNA sequences tailored to unique project specifications. While expressiveness provides flexibility for users, reconciling this flexibility constraint with optimal efficiency remains an open problem.

In this research, we use hardware accelerated algorithms to speed up the design process while preserving NUAD's expressiveness advantage. Furthermore, we extend the NUAD project by incorporating a constraint checker based on SIMD Smith-Waterman with affine gap penalties into the NUAD framework. Through this hardware acceleration resulting in a 10x speedup in constraint checking, NUAD offers enhanced efficiency and scalability for DNA sequence design. By combining the expressive power of NUAD with the computational prowess of SIMD Smith-Waterman, we can expedite the process of designing DNA sequences while upholding flexibility of user-defined constraints and specifications.

1 INTRODUCTION

At a high level, NUAD [1] uses a stochastic local search algorithm which is supported by various constraint checkers. Initially, this algorithm receives input specifications, which encompass measurable constraints like temperature, energy thresholds, and binding requirements. Additionally, the user sets a score threshold, defining a valid DNA strand as one with a score below this threshold.

To evaluate these constraints against the DNA sequences generated by the stochastic local search, NUAD employs eight constraint checkers. Some of these checkers have been developed in-house, while others are sourced from popular free energy model packages like NUPACK and ViennaRNA. Users have the flexibility to select which constraints they wish to apply, and they can even define custom constraints tailored to their specific DNA synthesis objectives.

Each constraint checker yields a score, which is then aggregated into a combined score. If this combined score falls below the user-defined threshold, the group of DNA strands satisfying the constraints is returned. Conversely, if the score exceeds the threshold, the stochastic process iterates to generate the next generation of DNA sequences, aiming to lower the score. Importantly, if an individual constraint's score surpasses the threshold, the remaining constraints are not executed, optimizing computational efficiency.

2 GOALS

Expanding on NUAD's stochastic local search algorithm, our focus centers on optimizing constraint evaluation, specifically in scoring DNA sequences. Inspired by the Smith-Waterman algorithm [7] used in RNAplex [8] by viennaRNA [4], we identified an opportunity to leverage SIMD (Single

Instruction, Multiple Data) to build a highly efficient constraint checker with Smith-Waterman as its underlying energy model.

In this project, we introduce a new constraint tailored for NUAD. Leveraging a C++ implementation of SIMD Smith-Waterman inspired by the SHRiMP aligner [6] as well as subsequent work by Dr. Thachuk, we fine-tuned coefficients to create a faster albeit less accurate version of RNAplex. Our improved algorithm and implementation seamlessly integrates with NUAD through dynamic linking and is optimized for inter-process communication efficiency.

One of the goals is to compare the performance of this new constraint against established tools like RNAplex and NUPACK. Through thorough testing and analysis, I showcase the effectiveness of my approach in accelerating DNA sequence design within the NUAD framework.

3 BACKGROUND IN SIMD

The genomic sequence local alignment paradigm, commonly referred to as local alignment, asks the following question: given two nucleic acid sequences and a list of scoring coefficients which describe similarity, what are the most mathematically similar pairs of segments? Landmark paper Identification of Common Molecular Subsequences [7] contained the Smith-Waterman algorithm, an efficient approach to solving the genomic sequence local alignment paradigm.

SIMD, short for Single Instruction Multiple Data, is a computing technique which breaks a single processor into multiple computing units in order to execute low-level operations in parallel. Realizing that the underlying inefficiencies in the Smith-Waterman algorithm could be resolved with SIMD, researchers separately developed and published versions of Smith-Waterman using SIMD known as anti-diagonal, sequential, and striped [2, 5, 10]. Each new variant introduced a new way of visually analyzing the algorithm, and each saw massive efficiency gains over its predecessors.

3.1 Anti-Diagonal

The first SIMD accelerated Smith-Waterman algorithm was devised by Andrzej Wozniak in 1997, now known as the anti-diagonal method. He utilized the intrinsic functions included in the extension set on the newly released Sun Enterprise 6000 [10]. Intrinsics, or intrinsic functions, allow programmers to access low-level assembly instructions without needing to use complex inline assembly conventions. Algorithmically, Wozniak realized that each diagonal in the dynamic programming table was dependent only on its previous two diagonals. By loading portions of the diagonals into SIMD registers, one can compute a portion of the next diagonal in a single operation, effectively parallelizing Smith-Waterman. While the anti-diagonal approach did not change the asymptotic time complexity of Smith-Waterman, it achieved a two times constant time speedup over the barebones algorithm.

3.2 Sequential

The sequential approach to SIMD Smith-Waterman takes advantage a new heuristic known as a SWAT optimization. The SWAT optimization exploits the fact that most cell values do not exceed the sum of the gap opening and extension penalties. Furthermore, if the beginning of a column does not exceed this sum, vertical state transitions will never result in a value that exceeds said threshold. This optimization gave way for a sequential approach to SIMD Smith-Waterman, aligning SIMD registers along the verticals, parallel to the query sequence, and evaluating diagonal and horizontal state transitions with less computational overhead. Rognes and Seeborg pioneered this sequential approach using intel's SSE instruction on a Pentium III processor, achieving an astonishing six times speedup on average [5].

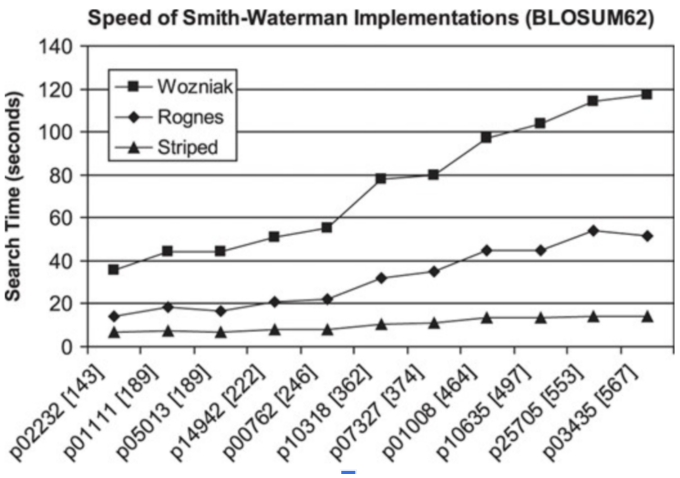

Fig. 1. comparing performance of algorithms applying SIMD (Michael Farrar, Striped Smith–Waterman speeds database searches six times over other SIMD implementations, Bioinformatics, Volume 23, Issue 2, January 2007, Pages 156–161, https://doi.org/10.1093/bioinformatics/btl582)

3.3 Striped

Like the Gotoh [3] modification of Smith-Waterman, Striped Smith-Waterman is a small yet complex and powerful modification of the Sequential approach. In striped Smith-Waterman, SIMD registers are aligned parallel to the query sequence, but instead of aligning registers on sequential vertical segments, registers occupy spots on a vertical line of the dynamic programming grid which are multiples of a specific offset away from each other. This approach achieves a further 6 fold speedup over Rognes and Seeborg's sequential approach, which equates to an impressive 50 to 60 fold improvement over naive Smith-Waterman [2].

Figure 1 shows the performance differences between Wozniak Rognes and Striped approaches to Smith-Waterman.

3.4 Use of SIMD DNA matching research in NUAD

Chris Thachuk has written a version of SIMD constraint check which can find complex free energy, and it was implemented by tuning the parameters of Smith Waterman to "fit the curve" of NUPACK's RNA to RNA tool. This was done almost completely empirically, changing values until data output resembled each other. Its main weakness is that it is less stringent than NUPACK, as it is a heuristic and cannot compete with its accuracy. We can use this algorithm as a first pass to rule out characteristic secondary structures, as any structure it rules out will also be ruled out by NUPACK. However if it passes the constraints, then we would still have to check with NUPACK's model.

4 DESIGN OF HARDWARE-ACCELERATED DNA SCORING

In designing our project, several critical decisions and implementations were made to ensure seamless integration of the SIMD Smith-Waterman constraint checker into NUAD. Firstly, the aforementioned SIMD-Smith Waterman code derived from the SHRiMP aligner employs the classic Wozniak anti-diagonal implementation but incorporates use-case specific optimizations that bring its efficiency on par with Farrar's striped Smith-Waterman algorithm in practice.

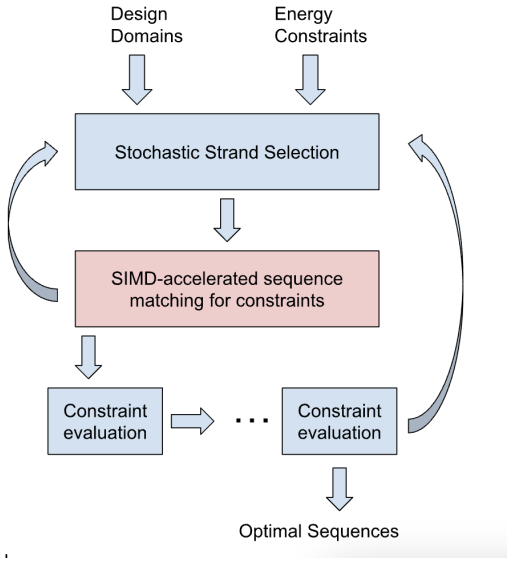

Fig. 2. SIMD-accelerated matching as a pre-filter to constraint evaluation in NUAD search algorithm

The entire user-facing interface was overhauled. Initially a standalone command-line application, we transformed it into a library, its aligner and member functions becoming callable from other programs. A pivotal step in this process was the creation of a Python wrapper atop the library using ctypes, enabling the Python-based NUAD to invoke the C++ functions of the SIMD Smith-Waterman code.

On the NUAD side, a new constraint, termed "lcs_simd_strand_pairs_constraint," was implemented within the constraints.py file. This constraint's input parameters mirrors the rest of the constraints, taking in user specifications and a list of pairs of DNA strands. Throughout the development process, various communication methods between Python and C++ were explored, ultimately settling on the use of numpy arrays passed directly to C++ functions via ctypes.

One crucial consideration was whether to statically or dynamically link the C++ library. Given NUAD's core value of flexibility, dynamic linking was chosen to eliminate the need for users and developers to compile the entire C++ library each time they used or tested NUAD. Dynamic linking being more memory and time efficient in comparison to static linking further justified this decision.

Figure 2 shows the design of the DNA search algorithm and the placement of SIMD-accelerated matching as a way to pre-filter DNA strands in the stochastic search process.

The SIMD Smith-Waterman constraint, by design, yields high scores, almost always surpassing those of other constraints, as it uses he least accurate complex free-energy model. Nonetheless, as elucidated in the introduction section, when a score exceeds a certain threshold, the stochastic search disregards the remaining constraints. Given the significant speed advantage of the SIMD Smith-Waterman constraint over others, it acts as a filtering agent, swiftly pruning the search space, speeding the overall stochastic process.

5 EVALUATING AVERAGE & WORST-CASE PERFORMANCE

Two sets of tests were designed to assess the efficacy of our SIMD Smith-Waterman implementation integrated into the NUAD framework. The first set comprised smaller, handcrafted tests, deliberately constructed to explore edge cases that could potentially challenge the efficiency of the algorithm. These scenarios included comparisons involving extremely disparate strand lengths, such as a

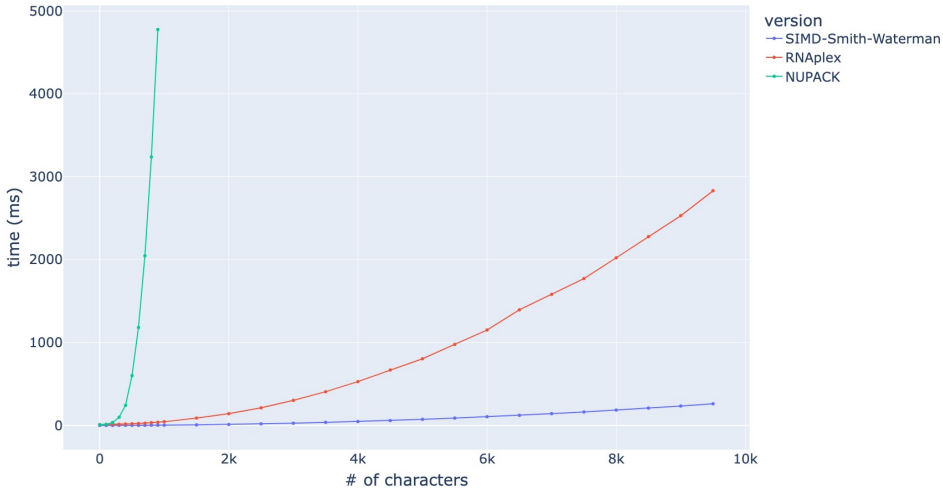

Fig. 3. Performance as a function of RNA strand length.

Fig. 4. Performance at typical RNA strand lengths.

length 5 strand juxtaposed with a length 1000 strand, or configurations deliberately chosen to induce the dynamic programming algorithm within the SIMD implementation to operate at its slowest possible pace. In contrast, the second set encompassed randomized data, with sequences extending up to 10,000 characters. Despite typical DNA strands falling within the range of 50 to 100 characters, the inclusion of these larger datasets allowed for the evaluation of performance under worst-case scenarios.

Figure 3 shows how processing time increases with RNA strand length for different models. Both SIMD Smith-Waterman and RNAplex demonstrate behavior aligning with $O(N^2)$ complexity, reflecting their underlying algorithms. Meanwhile, NUPACK's time usage aligns with $O(N^3)$. Notably, SIMD Smith-Waterman shows approximately 10 times greater efficiency on average compared to RNAplex.

Figure 4 examines model performance at typical RNA strand lengths of 50-100 characters. At these lengths, SIMD Smith-Waterman consistently outperforms RNAplex by about 5 times and NUPACK by about 10 times. This substantial efficiency gap underscores the advantages of hardware acceleration, particularly when dealing with sequences of moderate length.

6 PERFORMANCE OF SIMD-POWERED NUAD

In summary, the results depicted in the accompanying graph illuminate the superior performance of our SIMD implementation over both RNAplex and NUPACK. On average, our implementation exhibited a 5x improvement over RNAplex and an impressive 10x improvement over NUPACK, underscoring a significant enhancement in computational efficiency across all test sets. Remarkably, even in the worst-case scenario test cases deliberately crafted to stress the capabilities of the SIMD implementation, it consistently outperformed RNAplex and NUPACK by comparable factors to the overall average. Notably, as sequence length increased, NUPACK's performance deteriorated in line with its $O(N^3)$ complexity, contrasting with the $O(N^2)$ complexity of RNAduplex and the SIMD implementation. Meanwhile, RNAduplex consistently trailed behind by a constant factor of 10 on larger tests, further emphasizing the advantages of our SIMD Smith-Waterman approach for accelerating DNA sequence design within the NUAD environment.

7 CREDITS

I'd like to extend my deepest gratitude to my advisor Dr. David Doty, whose mentorship and support have been indispensable throughout this research endeavor. As the director of the Molecular Computing Lab at UC Davis, Dr. Doty generously shared his expertise and insights, always available to provide guidance, even on weekends. Furthermore, I am grateful to Dr. Doty for sponsoring this work through an NSF research grant, affording me the opportunity to contribute as an REU student. Furthermore, I express my appreciation to Dr. Chris Thachuk and Dr. David Gusfield for their invaluable contributions to this project.

8 CONCLUSION

In conclusion, our research demonstrates the significant impact of hardware acceleration on DNA sequence design within the NUAD framework. By leveraging SIMD Smith-Waterman, we achieved remarkable speed enhancements without sacrificing constraint flexibility. Even under worst-case scenarios, our implementation consistently outperformed established tools, highlighting the efficiency gains and computational advantages of our approach.

Moving forward, our work opens avenues for further exploration and optimization. One promising direction is the integration of our approach into broader computational frameworks. For instance, the Google Highways project [9] aims to provide intrinsics that enable code to run efficiently across diverse processor architectures. Adapting our SIMD implementation to utilize these intrinsics could extend its applicability beyond Intel Core series processors with SSE2 or higher.

Additionally, optimizing the communication between the C++ and Python components of NUAD remains an open question. Experimenting with different data structures and communication methods, such as arrays, lists, or numpy arrays, could further streamline the interaction between these components, improving overall performance.

Lastly, further testing and modifications may be necessary to ensure the relative accuracy of the SIMD Smith-Waterman constraint within the overall NUAD system. To this end, exploring reinforcement learning techniques to fine-tune the constraints of the Smith-Waterman algorithm presents an intriguing avenue for future research. By dynamically adjusting parameters based on feedback from the design process, we could potentially enhance the accuracy and effectiveness of constraint evaluation, ultimately facilitating more efficient DNA sequence design.

In summary, our research underscores the transformative potential of hardware acceleration in DNA sequence design and highlights exciting opportunities for future advancements in this field. By continuing to innovate and optimize computational tools like NUAD, we can empower researchers to unlock new possibilities in biotechnology.

REFERENCES

[1] David Doty. 1999. NUAD. (1999). https://github.com/UC-Davis-molecular-computing/nuad

[2] Michael Farrar. 2007. Striped Smith–Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics* 23, 2 (2007), 156–161.

[3] Osamu Gotoh. 1982. An improved algorithm for matching biological sequences. *Journal of molecular biology* 162, 3 (1982), 705–708.

[4] Ronny Lorenz, Stephan H Bernhart, Christian Höner zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. 2011. ViennaRNA Package 2.0. *Algorithms for molecular biology* 6 (2011), 1–14.

[5] Torbjørn Rognes and Erling Seeberg. 2000. Six-fold speed-up of Smith–Waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics* 16, 8 (2000), 699–706.

[6] Stephen M Rumble, Phil Lacroute, Adrian V Dalca, Marc Fiume, Arend Sidow, and Michael Brudno. 2009. SHRiMP: accurate mapping of short color-space reads. *PLoS computational biology* 5, 5 (2009), e1000386.

[7] Temple F Smith, Michael S Waterman, et al. 1981. Identification of common molecular subsequences. *Journal of molecular biology* 147, 1 (1981), 195–197.

[8] Hakim Tafer and Ivo L Hofacker. 2008. RNAplex: a fast tool for RNA–RNA interaction search. *Bioinformatics* 24, 22 (2008), 2657–2663.

[9] Jan Wassenberg. 2023. Google highway. (2023). https://github.com/google/highway

[10] Andrzej Wozniak. 1997. Using video-oriented instructions to speed up sequence comparison. *Bioinformatics* 13, 2 (1997), 145–150.