Understanding the Parallelizability of Gibbs Sampling for DNA Motif Finding

Abstract

This study introduces an optimized parallel version of the Gibbs sampling algorithm for DNA motif finding, implemented using CUDA. By comparing this with a sequential version across various input dimensions, significant performance improvements are observed in motif estimation for DNA sequences. Notably, the parallel algorithm did not scale equally well across all input dimensions. The findings underscore the importance of fine-grained performance measurements for parallelized bioinformatics algorithms.

Main Text

Introduction

Motif discovery plays a vital role in identification of Transcription Factor Binding Sites (TFBSs) that help in learning the mechanisms for regulation of gene expression.

A deeper understanding of DNA and other biological sequencing data is central to advancements in bioinformatics and computational biology. A critical aspect of this exploration is the identification of motifs – short, recurring patterns in DNA that are indicative of regulatory roles and biological functions. DNA motifs are believed to be significant in many biological functions, including DNA methylation [1], histone modifications [2], and transcription factor binding sites [3]. This is a long-standing problem in biology, and many algorithms have been proposed to find the shared motif across a series of DNA sequences [4, 5].

However, many motif-finding algorithms have trouble scaling to very large input sizes. This bottleneck necessitates a paradigm shift towards leveraging high-performance computing resources, particularly parallel computing frameworks. Some work has already been done to speed up motif finding using parallel computing Graphics Processing Units (GPUs), such as GPUmotif [6] and MIC-MEME [7]. However, these studies often only offer a coarse-grained understanding of the algorithm's scalability as the input size increases. Understanding what input dimensions yield the most benefit from parallelization and which are hardly improved is crucial for designing high-performance parallel algorithms.

Our study seeks to understand this by introducing benchmarked sequential and parallel implementations of a Gibbs sampling algorithm for DNA motif finding, varying different dimensions of the input size independently. We use the Gibbs sampling algorithm as it's a simple, popular, and well-studied algorithm for motif finding. Utilizing the Compute Unified Device Architecture (CUDA) for parallel processing on GPUs, we aim to understand the effects of parallelization on different aspects of the algorithm's input. We measure speedup across all input dimensions and report results below. We also release the code for this study for use in motif-finding tasks and for inspiration for developing efficient parallel bioinformatics algorithms.

Methods

Algorithm Implementation:

We developed a Gibbs sampling motif finding algorithm in the C++ programming language for both the parallel and sequential versions. This is a Markov Chain Monte Carlo (MCMC) algorithm, which uses stochastic updates to converge to the optimal motif. While exact solutions are not guaranteed, the algorithm leverages its speed to obtain the best approximate solution. The algorithm uses a Position Specific Scoring Matrix (PSSM) with added pseudocounts to allow for more accurate convergence.

At each iteration, the consensus motif is the PSSM-most probable motif. The score function is calculated from the sum of the Hamming distances between the consensus motif and each of the currently selected motifs in the PSSM. The score function is represented as:

$$\text{Score} = \sum_{i=1}^{N} \text{Hamming Distance}(\text{ Consensus, Motif[i] })$$

The sequential variant of the algorithm operated within a single-threaded environment. This algorithm served as a baseline for the performance of the parallelized version. The parallel algorithm preserved the semantics of the sequential algorithm while seeking to utilize an NVIDIA GPU acceleration through the CUDA API. This approach enabled the utilization of the CUDA kernels to harness the efficiency and parallel architecture of GPUs. Thus, highly parallelizable parts of the algorithm were replaced with CUDA kernels, allowing for the use of a GPU to run those parts of the algorithm.

Data Preparation:

To evaluate the parallelizability and efficiency of the algorithm, we generated random DNA sequences of varying lengths, where each base pair had an equal probability of being randomly chosen. It was not necessary for the DNA input sequences to resemble real DNA sequences for the purposes of measuring speedup.

Furthermore, to comprehensively evaluate the efficiency and validate the accuracy of our parallel algorithm, we conducted a comparative analysis against XX Gibbs sampling-based motif-discovery tools available on the internet. For this evaluation, we utilized the benchmark dataset curated by Tompa *et al* [7].

Assessing Parallelizability in Our Gibbs Sampling Algorithm:

In examining the parallelizability of our Gibbs sampling motif finding algorithm using synthetic sequences, we chose to vary each of the input dimensions, which consists of the different parameters of the algorith's input space (Table 1). The base values denote the initial values for the input dimensions under variation, while the default value represents the manually chosen value for the corresponding input dimension when it is not under examination. We chose to vary only one input dimension per experiment. For each iteration, we disregarded the output k-mer score and only considered the runtime, which is measured in milliseconds. Each run had six iterations, and the value of the varied parameter was calculated as follows:

$$\text{Parameter} = (\text{Base Value} \cdot 2^i),\ \forall i \in \mathbb{Z},\ 0 \leq i \leq 5$$

Table 1: Input parameters for our Gibbs sampling algorithm

Input	Base value	Default value	Command line flag	Description
Number of Iterations	200	400	-n	The number of times the algorithm "samples" new motifs before returning the best solution.
Length of Motif	5	10	-k	The length of the shared motif (as the number of DNA nucleotides) we are searching for in the input sequences.
Number of Sequences	200	400	N/A	The number of input DNA sequences.
Length of Sequence	250	500	N/A	Number of nucleotides of each input DNA sequence

This table presents the list of input parameters varied to test for parallelizability of the Gibbs sampling motif finding algorithm.

Performance Metrics:

The performance evaluation of the algorithm's parallelizability included comparing the runtime of both its parallel and sequential variant across various dimensions, and calculating the corresponding resulting speedup. For each set of input parameters, the speedup of the algorithm was calculated as follows:

$$\text{Speedup} = \frac{\text{Parallel Run Time}}{\text{Sequential Run Time}}$$

Experimental Environment:

The experimental setup utilized for conducting all benchmarking and analysis consisted of the following hardware components:

Table 3: Hardware Specification for Testing and Analysis

Component	Model/Specification
Processor (CPU)	Intel® Xeon® Processor E5-2603 (x2)

	Clock Speed: 1.80 GHz
	Cores: 4 (each)
	Cache: 10 MB Intel® Smart Cache (each)
Graphics Processing Unit (GPU)	NVIDIA GeForce GTX 1080 (x2)
	Graphics Clock: 1607 MHz
	Processor Clock: 1733 MHz
	CUDA Cores: 2560 (each)

This table presents the hardware specifications utilized in the experimental setup for conducting tests and analysis. The specifications include details on processors and graphics processing units(GPUs). For further details, readers are encouraged to consult the official manufacturer's manual.

The experimental setup utilized for conducting all benchmarking and analysis consisted of the following software components:

Table 4: Software Specification for Testing and Analysis

Software	Version
Compiler	NVIDIA CUDA Compiler(NVCC) version 11.6
Programming Language	C++
CUDA Toolkit	Version 11.6

This table presents the software specifications utilized in the experimental setup for conducting tests and analysis. The specifications include details on compiler, programming language, and CUDA toolkit.

Results

We developed and benchmarked both the parallel and sequential algorithms' performance across increasingly larger input data (Figure 1). Note that it is expected for the parallel algorithm to be slower than the sequential algorithm for small input sizes, as parallelization has inherent overhead. We can see that, as the input size increases, the parallel algorithm clearly outperforms the sequential algorithm across all dimensions except when scaling the number of sequences (Figure 1c). Interestingly, we see that scaling the length of the search motif has almost no effect on the parallel runtime even though the sequential runtime scales linearly in the motif length (Figure 1b). This demonstrates that a GPU-parallelized Gibbs sampling algorithm might be ideal for finding long DNA sequence motifs. This also underscores the power of parallel computing on GPUs, showing that across specific input dimensions, we can increase the input size with almost no effect on the runtime.

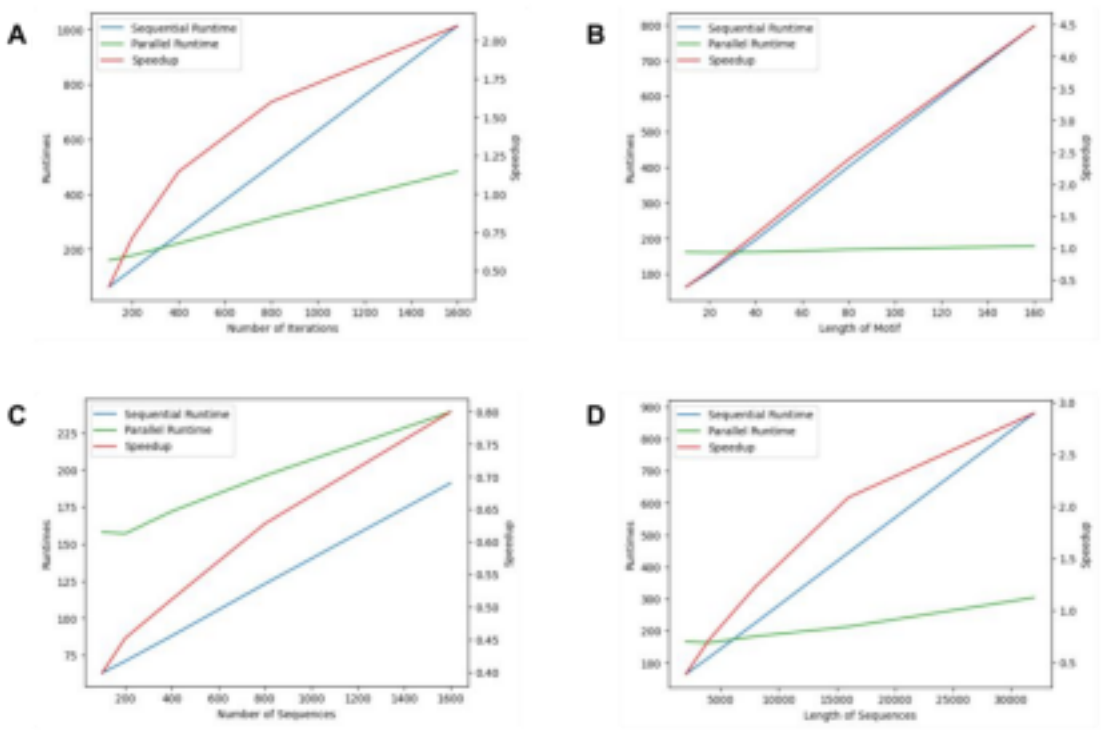

Figure 1: Graph showing both algorithms' runtimes (rounded to the nearest millisecond) and their associated speedups. A) shows the runtimes and speedup as we increase the algorithm's sampling iterations. B) shows the runtimes and speedup as we increase the length of the search motif, where the length is measured in base pairs. C) shows the runtimes and speedup as we increase the number of input DNA sequences. D) shows the runtimes and speedup as we increase the length of each sequence, where the length of each sequence is measured in base pairs.

To test if there was a speedup for a larger number of sequences, another experiment using 10,000 input sequences was run. The results show that the parallel algorithm is more performant than the sequential algorithm for many sequences, achieving a modest speedup (Table 1).

Number of Sequences	Sequential Runtime	Parallel Runtime	Speedup
10,000	931	622	1.50

Table 1: Table showing the runtimes (rounded to the nearest millisecond) and speedup of both algorithms with many input sequences. Note that the other parameters are the base values listed above.

Overall, the results of these experiments show that the parallel algorithm achieves a speedup across all

input dimensions. However, the performance increase varies greatly depending on which dimension of the input increases, with drastically varying results. This demonstrates that strategic parallelization of bioinformatics algorithms can yield significant performance benefits, but these performance benefits may be much stronger for certain classes of inputs.

Discussion:

Parallelizing bioinformatics algorithms is a promising avenue for the field, as many bioinformatics tasks must cope with bigger datasets while still running in a reasonable amount of time. However, certain problems may not see any benefit from parallelization, while other problems may see large benefits. This study underscores the importance of characterizing the performance landscape of a parallel algorithm, as it may have unintuitive speedup dynamics.

One limitation of this work is that it only studies one motif-finding algorithm. There are many other motif finding algorithms proposed in the literature that may also be effective targets for parallelization. Future work could include parallelizing other popular motif finding algorithms to observe algorithm-specific benefits derived from parallelization.

Other future work would include further optimizing the GPU code for the motif-finding task. The GPU code used in this study is general, allowing it to achieve performance benefits on a variety of GPUs. However, there are differences in GPU architectures that may allow for a better speedup using device-specific tweaks. Additionally, the kernel launches themselves are highly parallelizable; utilizing NVIDIA streams could allow many Gibbs sampling processes to run at once. Exploring how many processes can be effectively run at once is a promising avenue for further high-efficiency motif finding algorithms.

Statement: During the preparation of this work the author(s) used [NAME TOOL / SERVICE] in order to [REASON]. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References

1. Mengchi Wang, Kai Zhang, Vu Ngo, Chengyu Liu, Shicai Fan, John W Whitaker, Yue Chen, Rizi Ai, Zhao Chen, Jun Wang, Lina Zheng, Wei Wang, Identification of DNA motifs that regulate DNA methylation, *Nucleic Acids Research*, Volume 47, Issue 13, 26 July 2019, Pages 6753–6768, https://doi.org/10.1093/nar/gkz483
2. Ngo, V., Chen, Z., Zhang, K., Whitaker, J.W., Wang, M., Wang, W., 2019. Epigenomic analysis reveals DNA motifs regulating histone modifications in human and mouse. Proceedings of the National Academy of Sciences 116, 3668–3677. https://doi.org/10.1073/pnas.1813565116.
3. Mario Pujato, Fabien Kieken, Amanda A. Skiles, Nikos Tapinos, Andras Fiser, Prediction of DNA binding motifs from 3D models of transcription factors; identifying TLX3 regulated genes, *Nucleic Acids Research*, Volume 42, Issue 22, 16 December 2014, Pages 13500– 13512, https://doi.org/10.1093/nar/gku1228D
4. Das, M.K., Dai, HK. A survey of DNA motif finding algorithms. BMC Bioinformatics 8 (Suppl 7), S21 (2007). https://doi.org/10.1186/1471-2105-8-S7-S21
5. Hashim FA, Mabrouk MS, Al-Atabany W. Review of Different Sequence Motif Finding Algorithms. Avicenna J Med Biotechnol. 2019 Apr-Jun;11(2):130-148. PMID: 31057715; PMCID: PMC6490410.

6. Zandevakili, P., Hu, M. & Qin, Z. GPUmotif: An Ultra-Fast and Energy-Efficient Motif Analysis Program Using Graphics Processing Units. PLOS ONE 7, e36865 (2012). 7. Peng, S., Cheng, M., Huang, K. et al. Efficient computation of motif discovery on Intel Many Integrated Core (MIC) Architecture. BMC Bioinformatics 19 (Suppl 9), 282 (2018). https://doi.org/10.1186/s12859-018-2276-1

7. Tompa, M., Li, N., Bailey, T. *et al.* Assessing computational tools for the discovery of transcription factor binding sites. *Nat Biotechnol* 23, 137–144 (2005). https://doi.org/10.1038/nbt1053