Optimization of Exact Algorithms for Planted $(l,d)$-Motif Problem

Andrew Petersen (Loyola Marymount University)
Edward Felekey (University of Connecticut)
2013 NSF Bio-Grid REU Research Fellows
University of Connecticut: Department of Computer Science and Engineering

Motif search – the process of locating short patterns that occur frequently within a set of sequences – applies directly to the field of computational biology. Patterns in DNA and polypeptide sequences reveal connections that might otherwise be too difficult to find without computational tools. These motifs are relevant to biological research, particularly to the complex regulatory networks that occur within the nucleus of a cell. New transcription factor binding sites can be found with motif search algorithms, which can help form a more complete picture of how non-coding DNA works. These new understandings in turn help the world of medicine develop better methods to treat disease. However, exact algorithms to find these motifs are computationally intensive and run in exponential time, so it is of interest to find more efficient ways to implement this search to expand the number of cases that are reasonably computable.

The purpose of this project is to efficiently implement the exact algorithms PMS1, PMS2, and PMS3 (as developed by Rajasekaran et al), which solve the Planted $(l,d)$-Motif Problem. This problem is defined as follows: given $t$ sequences of length $n$, alphabet $\Sigma$, desired Motif length $l$, and Hamming distance $d$, find a Motif $M$ of length $l$ that has a variant in every input sequence. PMS1, which has already been implemented, runs in $O(tn(l)(|\Sigma|^{d+1})w)$ time (where $w$ is the word length of the computer) and requires a like amount of space. While the asymptotic growth cannot be changed without fundamentally changing the overall algorithm, practical performance can be improved by using well-constructed data structures, efficient algorithms, and low-level operations.

PMS1 and PMS2 algorithms have been implemented and are currently being refined, and work on PMS3 will follow. In these implementations, integers are used to represent the sequences, allowing for increased speed in isolation and manipulation of characters when compared to string representations. Currently, these implementations spend most of their time sorting lists of candidate motifs, while the other phases of the algorithm run much more quickly. Going forward, faster implementations of a radix sort algorithm will be employed to reduce processing time. However, methods to reduce the amount of sorting needed or eliminate sorting altogether will also be investigated. Such methods include generating candidate motifs in a predictable order or using a hash set to organize motifs without sorting. Also, these implementations hold enormous lists of candidate motifs in memory at once. To mitigate this, as lists of motifs are generated, only their intersection will be stored, eliminating the need to store the entire merged list and greatly reducing the memory requirements of the program. Run time and memory usage data from thorough tests using combinations of these proposed alterations with various input parameters will be collected and analyzed to benchmark the performance of these implementations.

After benchmarks have been made, a web interface will be created to allow others to make use of these computational tools. Online users will be able to input data about the size of their computation (number of sequences, sequence length, length of motif, Hamming distance, etc.) and receive an estimate of how long it will take to process. Then the user may choose to run the program on their specified input, or decline if processing would exceed a reasonable amount of time.

This work is not only applicable to implementations of other motif search algorithms, but also to implementations of other computational biology algorithms. The techniques explored in this work could be used in any context requiring fast and efficient operations on large quantities of long strings.