Poster H-68

A Comparison a New Genetic Algorithm With Existing Methods For Determining a Consensus Sequence

Authors:
Josh W Gilkerson (*University of Kentucky, Dept. of Computer Science*)
Dr. Jerzy W. Jaromczyk (*University of Kentucky, Dept. of Computer Science*)

Short Abstract: We present two original methods for finding consensus sequences and compare them with existing algorithms. Experimental tests are performed on both synthetic data and sequences taken from biological sources. The algorithms construct high quality consensus sequences -- superior to other methods -- in particular for repeat elements in the Magnaporthe grisea genome.

Long Abstract:

BACKGROUND

A consensus sequence attempts to capture the similarities among a family of sequences. Applications of consensus sequences and related constructs such as multiple alignments are pervasive in the field of bioinformatics. While the use of consensus sequences has been disparaged, an algorithm that determines high quality consensus sequences is nonetheless useful. In particular, a high quality consensus sequence can be used to efficiently construct a multiple alignment and consequently, a sequence logo or sequence walker. We present two approaches which have not been found in the existing literature.

This work was originally inspired by our work on RepeatAssembler, software that finds repeated elements in genomic data. These elements are repeated throughout the genome, possibly with slight differences. This creates a need to report the set of identified repeats in a concise manner, which led to our exploration of consensus sequences.

OUR APPROACH

The first novel method is an application of the genetic algorithm scheme (GAS), which has shown success when applied to other NP-Hard optimization problems. The GAS attempts to simulate the process of evolution through survival of the fittest. An initial population of candidate solutions is chosen. The population is then repeatedly subjected to a number of operations including analogs of sexual reproduction, mutation, education, and death. We implement multiple methods for all of these operations which the user may be select at runtime. The expected running time for our implementation is O(k^2 * l^2 * i), where k is the number of sequences in the family, n is the length of those sequences and i is the number of iterations over which the algorithm is run. One additionally attractive feature of this algorithm is that the user can decide how to make the trade-off between quality and computational resources.

The other new approach is a local search with deterministically greedy choices. A single

candidate sequence is iteratively improved by choosing the best from a number of alternatives at each iteration. Greedy algorithms tend to reach locally optimal solutions which may be far from the actual optimum. However, in this case it has shown itself to be surprisingly effective. The worst case running time for this algorithm is O(k * l^5), however this is rarely the case. Empirically, the running time has been closer to O(k * l^3).

EXPERIMENTAL ANALYSIS

We present and compare an array of methods for approximating consensus sequences. We also use the sequences generated by these methods to induce a multiple alignment and compare these results with popular multiple alignment algorithms. In addition to the above methods, we explore three existing approaches for comparison. These methods vary significantly in both accuracy and speed.

One method is the use of a multiple alignment to form a consensus sequence. A multiple alignment is generated using any of the well-known methods. The consensus sequence is then formed by taking a consensus character from each column. Both the speed and accuracy of this method depend on the method used to generate the multiple alignment. For example, using EMBL's clustalw is very fast, but the accuracy is often disappointing.

Another method is the application of simulated annealing (SA) described by Keith et al. This method is similar to the greedy algorithm mentioned above, but at each iteration the choice is made non-deterministically, but favoring better alternatives. This allows the iteration to escape local optima. The worst case running time for the SA algorithm is O(k * l^3 * i).

A third method, described by Gusfield as the center string method, is to consider each of the input sequences as a candidate consensus sequence and choose the best. This is a simple, yet surprisingly effective method. It is very fast, running in O(k * l^2) time, and provides accuracy guarantees.

RESULTS

We present results from all of these methods when applied to both synthetic and biological data. Additionally, we explore the quality of the multiple alignments induced by the resulting consensus sequences when compared to existing multiple alignment algorithms. For biological data, examples of repeated elements from the Magnaporthe grisea genome are used. The synthetic data is generated by mutating a random sequence with probabilities corresponding to the weights used for scoring.

Our results show that the greedy algorithm produces unexpectedly accurate results. clustalw is very fast, but the accuracy is not as good as from other methods. The randomized algorithms (SA and GAS) outperform the fast methods and when allowed to run for a comparable amount of time are equal to or better than the greedy method.

REFERENCES

Gusfield, Dan. Algorithms on Strings, Trees, and Sequences. Cambridge University Press, New York, NY. 1997.

Keith, Jonathan M., Peter Adams, Darryn Bryant, Dirk P. Kroese, Keith R. Mitchelson, Duncan A. E. Cochran, and Gita H. Lala. ``Simulated Annealing Algorithm for Finding Consensus Sequences.'' Bioinformatics. 18(2002):1495-9.