Optimization of Computations Used in Information Theory Applied to Base Pair Analysis

UMBC REU Site: Interdisciplinary Program in High Performance Computing
Team members: Andrew Coates[1], Alexey Ilchenko[2]
Faculty Mentors: Matthias K. Gobbert[1], Nagaraj K. Neerchal[1] Client: Patrick O'Neill[3], Ivan Erill[3]
[1]Mathematics and Statistics, UMBC [2]Mathematics, Case Western [3]Biology, UMBC

Problem Statement

Biologists use Information Theory as a method of predicting where proteins will bind to DNA. They specifically use entropy and information content. Expected entropy is calculated in order to determine information content. Its calculation is very time consuming when done exhaustively for medium and large sample sizes, n, and also runs into memory problems. We wanted to develop an improved algorithm to compute expected entropy. The main range of interest is around 20 to 70 samples.

Base Pairs

There are four chemicals found at binding sites in DNA: Adenine (A), Thymine (T), Cytosine (C), Guanine (G). Sets of samples can be represented by motifs:

Sample	Position			
	1	2	3	4
1	A	T	C	G
2	T	G	G	G
3	T	C	G	G
4	A	T	C	G
5	C	G	G	T
6	A	A	G	C

Information Theory

- Entropy measures uncertainty: $E(x) = -\sum p(x) \log_2 p(x)$.
- Information Content (IC) is the loss of uncertainty, or the gaining of information.
- $IC = E(H_n) - H_{x|y}$ where $E(H_n)$ is expected entropy of n samples and $H_{x|y}$ is entropy of a given binding site.

$E(H_n)$ Computed in $O(4^n)$

- Computing $E(H_n)$ is an expensive computation: $E(H_n) = -\sum_{s \in \Delta^n} H(s)p(s)$
- Δ^n is the cross between Δ with itself n times.
- $H(s)$ is computed with relative frequencies of s.
- $p(s)$ the product of genomic probabilities to respective exponents
- This operation is of order $O(4^n)$.

The $O(n^3)$ Algorithm

- Partitions of n: all natural number combinations adding up to n.
- Compositions of n: all permutations of the partitions of n.
- We strictly use length four compositions in the form (N_A , N_T , N_C , N_G).
  - Multinomial coefficient compensates for lost permutations.

$$E(H_n) = -\sum_{v \in C} \binom{n}{v_1, v_2, v_3, v_4} H(v)p(v)$$

Coding Languages

- Exhaustive python code generated the strings of each item in Δ^n and then iterated through each string to compute $E(H_n)$. Order $O(4^n)$.
- MATLAB used $O(n^3)$ algorithm.
- C used $O(n^3)$ algorithm.

Results and Conclusions

	Run times in seconds		
n	python $O(4^n)$	MATLAB $O(n^3)$	C $O(n^3)$
1	< 0.0001	0.0007	0.0001
2	0.0003	0.0015	0.0002
4	0.0046	0.0053	0.0002
8	0.7603	0.0231	0.0008
13	863.8485	0.0576	0.0012
16	O.M	0.0963	0.0018
32	O.M	0.6016	0.0103
64	O.M	5.0046	0.0877
128	O.M	46.6464	0.5354
256	O.M	498.3383	3.6238

O.M = Out of Memory

- The MATLAB and C $O(n^3)$ code ran faster and scaled better with n than python $O(4^n)$ code.
- All values in range 20 to 70 can be solved exactly under 1 second.

References and Acknowledgements

Work done by Andrew Coates (coates3@umbc.edu) and Alexey Ilchenko (axi48@case.edu).
For more information see Technical Report HPCF–2011–13 at `www.umbc.edu/hpcf` > Publications.

Acknowledgment: This research was conducted during Summer 2011 in the NSF-funded REU Site: Interdisciplinary Program in High Performance Computing (`www.umbc.edu/hpcreu`) in the Department of Mathematics and Statistics at UMBC. The cluster tara in the UMBC High Performance Computing Facility (`www.umbc.edu/hpcf`) is partially funded by the National Science Foundation.