15th Symposium on Computer Architecture and High Performance Computing

November 10 to 12 - São Paulo, SP

Comparison of Genomes using High-Performance Parallel Computing

N. F. Almeida Jr

Universidade Federal de Mato Grosso do Sul

C. E. R. Alves

Univsidade São Judas Tadeu

E. N. Cáceres

Universidade Federal de Mato Grosso do Sul

S. W. Song

Universidade de São Paulo

Comparison of Entire Genomes

- Comparison of genomes is useful to investigate common functionalities of the corresp. organisms
- Our purpose is twofold
  - Use parallel computing so that more expensive alignment methods (dynamice programming) can be used.
  - Locate and compare not only homologous genes, but also compare the regions between corresponding homologous genes.
- As example, we compare
  - *Xanthomonas axonopodis* pv. *citri* with 5,175,554 base pairs and 4,313 protein-coding genes
  - *Xanthomonas campestris* pv. *campestris* with 5,076,187 base pairs and 4,182 protein-coding genes.

Motivations and Previous Works

- **Homology: two genes share a common evolutionary past.**
- **Often similarity between two DNA or amino-acid sequences may infer homology.**
- **Homology in turn may determine function.**

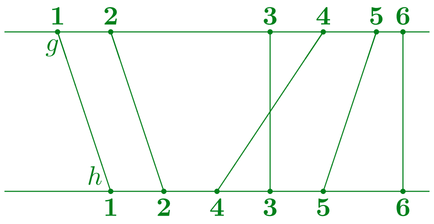
Thus: Similarity → homology → function

Rasera, Setubal, Almeida et al. [2002] compare the whole genomes of *Xanthomonas axonopodis* pv. *citri* and *Xanthomonas campestris* pv. *campestris* and conclude both share more than 80% of the genes.

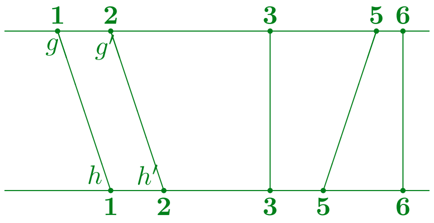
Comparison Strategy - Main Ideas

Given two genomes G and H and their gene locations:

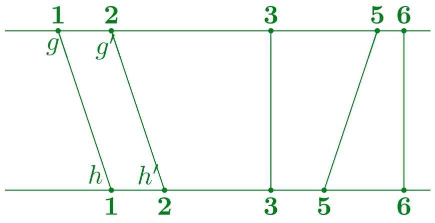
1. **Find and label pairs of homologous genes.**



2. **Find the non-crossing pairs of homologous genes.**



Comparison Strategy (continued)



3. Align each pair of homologous genes.

4. Align each pair of intergenic regions (e.g.$[g, g']$ and $[h, h']$).

5. Join all alignments.

Comparison Strategy - Details

1: *Find pairs of the homologous genes:*
For all g of G, obtain h of H such that DP-score(g, h) = max { DP-score(g, w) for all w of H }

2: *Label the homologous genes of G*:
Label the homologous genes of G as $1, 2, \ldots, m$ in the same order as their positions in the genome G.
Let *LabelG* denote the sequence of labels obtained in this step.

3: *Label the corresponding homologous genes of H*:
For all pairs of homologous genes (g, h), g of G, h of H, label gene h with the same label of g.
Let *LabelH* denote the sequence of labels obtained in this step.

4: *Find the non-crossing pairs of homologous genes*:
Obtain the LCS(*LabelG*, *LabelH*). *the LCS obtained contains only the non-crossing pairs*

5: *Align each pair of homologous genes*:
For each non crossing homologous pair (g, h) do DP-align(g, h).

6: *Align each pair of intergenic regions*:
For each intergenic region $[g, g']$, where $[g, g']$ of G are two consecutive genes of the LCS, obtain the corresponding intergenic region $[h, h']$ in H and do DP-align$([g, g'], [h, h'])$.

7: *Join all the alignments*:
Concatenate the alignments of the homologous genes and the intergenic regions, in the same order they appear in the genomes.

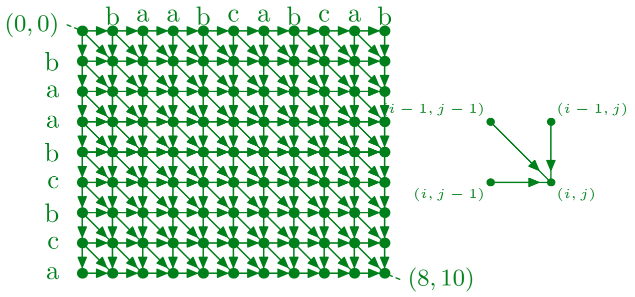
Computing Similarity of Two Strings

A simple example of string alignment:

A	a	c	t	t	c	a	–	t	
C	a	t	t	c	–	a	c	g	
Score	1	0	1	0	0	1	0	0	3

A	a	c	t	t	c	a	–	t	
C	a	–	t	t	c	a	c	g	
Score	1	0	1	1	1	1	0	0	5

Using dynamic programming (gives better quality alignments):



The Parallel Solution

- Finding homologus pairs (the most time consuming phase): compare all the genes of one genome with all the genes of another: more than 18 million alignments by dynamic programming.

  Two types of parallelisms are used:

  – Master distributes the alignment tasks to slave processors.

  – When the lengths of the sequences to be aligned exceed 5,000 base pairs, parallel dynamic programming is used.

- Finding the non-crossing homologous gene pairs: We used a parallel LCS (longest common subsequence) algorithm. (Could have used LIS - longest increasing subsequence algorithm.)

The Parallel Platform Used

- 64-node Beowulf cluster - low cost microcomputers with 256MB RAM, 256MB swap memory, CPU Intel Pentium III 448.956 MHz, 512KB cache.
- 100 Mb fast-Ethernet switch.
- Code in standard ANSI C and LAM-MPI Version 6.5.6.

Preliminary Implementation Results

- Finding homologus pairs (most time consuming):

  Sequential solution using Blast and EGG: 3 hours.

  Parallel solution using dynamic programming: 1 hour 15 minutes.

- Finding non-crossing pairs (surely not the dominant step):

  Sequential solution using Blast and EGG: not available.

  Parallel solution using dynamic programming: 20 seconds.

Conclusion

We compared the whole genomes of two organisms:

- Exploited parallelism in two ways:

  Standard master-slave approach to distribute comparison tasks (sequential dynamice programming) to slave processors.

  To compute the similarity between two sequences, whenever the sequences are longer than 5,000 base pairs, we used parallel dynamic programming.

- The gain does not seem to be so significant, however we used a dynamic programming approach that gives better quality results.

- Our comparison strategy also compares the intergenic regions between two consecutive homologous genes in each genome. The relevance of this in a biological viewpoint is yet to be investigated.