Synteny Determination: Processes, Problems, and Potentials

The past several decades have seen a revolution in techniques for molecular biology, and sequencing techniques are no exception to this. Building upon Sanger's 1977 sequencing method which helped to usher in the modern era of molecular biology, advances in sequencing such as shotgun or expressed sequence tag sequencing have greatly improved our ability to generate genomic data. The molecular biology side of genomics, however, only tells half of the story. In parallel with biology advances was the growth in computing power that made the analysis of the sequencing data generated possible. These advances in molecular biology and computer technology led the completion of several genomes, starting with the first free-living organism, *H. influenza* (Fleischmann et al., 1995), through the completed Human Genome six years later (Lander et al., 2001; Venter et al., 2001), and on towards the current goal of a $1000 personal genome.

The completion of multiple genomes has made possible whole genome or whole chromosome comparisons. Early on, it was learned that gene order between genomes is conserved. This concept is referred to as synteny, and it results from the evolutionary relationship between organisms. Of course, this conservation is not perfect. For one-chromosome events, inversions are a particularly common mutation, while for multichromosomal events, reversals, translocations, fissions, and fusions can all occur. These, combined with gene duplications and deletions among other processes, can explain how synteny between organisms can decrease with evolutionary distance.

Description of synteny and whole genome comparisons is not simply an academic exercise. It has many practical ramifications and can lead to important biological understandings. As was implied from the discussion above, synteny expresses phylogenetic relationships and

what steps were necessary to "convert" one genome into another on a macroscale (chromosomal level rather than nucleotide level). While comparisons between genomes can give synteny, the same comparisons within a single genome can also identify paralogs. Synteny can help with gene identification and genome assembly. Identifying regions of homology between a genome currently being sequenced and one already finished can speed up the completion of the first genome (Pop et al., 2004). Similarly, comparing two genomes may lead to the identification of new genes in each, whether through the identification of conserved—and hence possibly functional—unknown regions or through a previously known gene product of one genome being directly found in the other (Tomii and Kanehisa, 1998; Kihara and Kanehisa, 2000). Comparative genomics can also help us understand biological function. A knowledge of gene duplication events can shed light onto redundancy in genomes and its functional significance. Additionally, genes that are close to each other on a chromosome may have coordinated regulation, and observing changes within these regulatory blocks with time can potentially lead to new understandings of their functionality. Comparative genomics may perhaps allow us to better understand qualities of clinical interest, such as virulence or drug-resistance, too. At its core, the ability to make accurate and practical synteny maps between genomes is a prerequisite for making sense of the deluge of information currently being generated by sequencing efforts.

The determination of synteny presents a number of difficulties not found in traditional sequence comparison methods such as Needleman and Wunsch (1970) and Smith and Waterman (1981). Perhaps chief among them is time. Simply put, genomes are very large. The human genome, for instance, is 3.2×10^{12} bases. To align these by conventional dynamic programming algorithms with present computing technology would be prohibitively slow. To be practical in modern research, alignments must be able to be made on the order of minutes, hours, or days,

rather than weeks or months. As genome assemblies are often updated with more accurate sequencing data, a method of efficiently carrying this updating through to the alignments already done with other genomes is necessary, requiring fast large-scale alignment algorithms.

There are other problems of varying degrees of difficulty in enacting whole genome comparisons besides just their high computational demands. Some allowance must be made for organisms having genomes of different sizes and with different numbers of chromosomes, as well as for genes being able to be found in any orientation on a chromosome. Also, it is not nearly as easy to accomplish and meaningfully interpret a comparison of closely related genomes versus distantly related genomes; it is easier to draw relevant lessons from a human vs. mouse comparison rather than a human vs. *E. coli* comparison. It must also be kept in mind what exactly the goal is of a particular genome alignment. Different algorithm designs are best suited for studying macroscale differences (i.e. an inversion of a portion of a chromosome) than for studying microscale differences (a point mutation at a particular residue). Likewise, some decision must be made about how to control for differences in the noncoding regions of the genome. Different genomes can have vastly different amounts of repetitive DNA, leading to very different gene densities. If one is interested only in relative gene order, a large insertion of selfish DNA into a noncoding region might not be particularly relevant to the synteny between two genomes. Like with traditional sequence alignments, repetitive DNA can lead to erroneous alignments, and so there is often a need to mask it in whole genome comparisons. Finally, it should be noted that the degree of synteny (macroscale conservation of genome arrangement) and degree of homology (microscale conservation of a particular sequence) are different concepts, although they often go hand-in-hand. As a side note, how to visualize syntenic

relationships so they can be readily understandable by a researcher is far from a trivial task, but is outside the scope of this paper (Hunt et al., 2004).

One of the largest divisions between different approaches in large-scale comparative genomics is whether they are rooted in using protein or DNA to arrive at the synteny between the genomes. Each approach has its advantages and disadvantages. Protein is more conserved than DNA at a sequence level (and even more so if structure could be examined effectively on a genome-wide scale). At far enough evolutionary distances, conservation will be able to be detected at the protein level but not at the DNA level. Looking at protein for genome comparisons, however, requires annotated genomes, and thus can miss un-annotated regions of homology. DNA comparisons, in contrast, do not require annotations. This means that DNA comparisons can see conservation among non-protein coding regions, such as among promoters, enhancers, rRNA, small non-coding RNA, and other such genome features. DNA comparisons, as mentioned above, may lead to the discovery of un-annotated genes.

There are many algorithms to compare genomic-sized sequences; each one uses a different method of addressing the problems above and has its own weakness and strengths. As might be guessed from the preceding discussion, no one algorithm can effectively solve every problem in synteny discovery. Below, several of these algorithms will be examined. This is not meant to be a comprehensive listing of algorithms nor a complete descriptions of any particular algorithm, but rather, to understand general approaches to effectively comparing genomes.

A number of researchers have applied lessons first learned in the BLAST (Altschul et al., 1990) and FASTA (Pearson, 2000) algorithms to identify regions of synteny. Schwartz et al. (2003) were particularly interested in aligning neutrally evolving regions, and thus they wanted a particularly sensitive algorithm. As a result, they used a modified BLASTZ. BLASTZ

(Schwartz et al., 2000) is based on Gapped BLAST (Altschul et al., 1997) with several changes, including the removal of lineage-specific interspersed repeats, the requirement for matching regions to occur in the same order and orientation, and a scoring system that makes it harder for regions of biased nucleotide content to trigger a gapped alignment. BLASTZ was also modified to increase speed by masking regions to which many segments of the other sequence match and by looking at a larger window of 12 positions instead of the 8 previously used by BLASTZ, but allowing for one transition. This modified BLASTZ was indeed sensitive and specific—more sensitive than PatternHunter or BLAT to be discussed below—as was their goal. However, it came at a price. Aligning 2.8 GB of human sequence with 2.5 GB of mouse sequence took 481 days of Pentium III CPU time. This is prohibitively slow for many applications. Also, their modifications will throw out some potentially interesting regions of gene duplication. Thus, while sensitive and good at recognizing homology in non-coding regions, Schwartz et al.'s algorithm is very slow and misses some alignments.

Ma et al. (2002) created an algorithm called PatternHunter. At its heart is a clever idea: while BLAST uses a seed requiring the continuous alignment of multiple residues, PatternHunter uses a discontinuous seed. For example, PatternHunter uses a seed of 111010010100110111, where a 1 represents an exact match and 0 does not require a match. This method simultaneously gives a higher probability of a hit in homologous regions while having a lower frequency of random hits since the different positions function more as independent events. Hits are then looked up in a hashtable to find diagonals and extended in a greedy fashion, allowing for gaps. Compared to BLASTN, PatternHunter is 20 times faster, uses one-tenth the memory, and gives better results. A comparison of the human genome with a total of 9×10^{12} base pairs from unassembled mouse genome reads took 20 or 80 Pentium III CPU-days, depending on the

sensitivity used. While still slow, this represents a drastic improvement over previous BLAST methods. PatternHunter was also later updated to improve performance (Li et al., 2004).

PatternHunter was employed by Pevzner and Tesler (2003) to generate a synteny map with the goal of determining the most parsimonious human-mouse rearrangement scenario. Using bidirectional best local similarities, or anchors, from PatternHunter as a start, they used GRIMM (Teseler, 2002) to generate synteny blocks. GRIMM (Genome Rearrangements In Man and Mouse) forms an anchor graph whose vertex set is the anchor set, and if the vertices are close enough together based upon the parameters selected, they are connected by an edge to form a cluster. Small clusters are deleted since they are likely to be false orthologs. The clusters are then outputted as synteny blocks. They used the common approach of concatenating the chromosomes, and gave a sign to gene regions to mark orientation. While PatternHunter is still relatively slow and the parameters must be set correctly for GRIMM, Pevzner and Tesler showed how raw sequence alignments could be processed to form synteny maps.

SSAHA (Sequence Search and Alignment by Hashing Algorithm) (Ning et al., 2001) and BLAT (BLAST-Like Alignment Tool) (Kent, 2002) work by largely similar principles. Both of them construct an index of k-tuples with their corresponding positions in the genome, which can be screened against the query sequence. Each algorithm then has its own filters, such as SSAHA ignoring any word with a frequency above a set point since it is likely to be repetitive DNA. SSAHA finds runs of hits that lie along a diagonal, allowing for some gaps, to find regions of homology, while BLAT clumps multiple perfect hits within a gap limit and then extends these larger alignments. SSAHA is 3-4 orders of magnitude faster than BLAST or FASTA, although a direct comparison of their sensitivities was not done. BLAT, which can work with both nucleotide and protein alignments, took 16,300 CPU hours (Pentium III) to align the translated

human and mouse genomes. There are some other differences between them. SSAHA cannot unsplice mRNA as BLAT can, and it uses a single perfect match as its seed. Also, BLAT's efficiency decreases when the percent nucleotide identity between the genomes is below 90%, illustrating one of its weaknesses.

Suffix trees were the approach taken by Delcher et al. to reduce calculation time for whole genome comparisons in the MUMmer (1999) and MUMmer 2 (2002) algorithms. MUMmer 2, which stands for maximal unique match, concatenates all protein sequences of a chromosome (or translates the DNA sequence in all six frames first if the input is a nucleotide sequence), finds all maximal unique matches, and clusters those that are on similar diagonals by DNA position on an alignment matrix. The matches made are unique in that they occur once in each genome and are maximal in that they cannot be extended while still exactly matching. A filter is used on the translated genome to remove excessive numbers of stop codons, and the reading frames are picked to maximize protein homology. Notable, this approach will miss non-coding regions of DNA. MUMmer 2 improved upon the original MUMmer in speed and memory by a factor of three by storing the suffix tree for only one genome and streaming the second genome by it to find exact matches, in addition to other modifications to the suffix tree (Kurtz, 1999) and clustering method. These changes help to alleviate earlier criticisms about MUMmer's excessive memory requirements. In terms of speed, MUMmer 2 can align two bacterial genomes in under a minute on a standard desktop.

Several algorithms start with all vs. all BLASTP searches as input, including ADHoRe (Vandepoele et al., 2002), DiagHunter (Cannon et al., 2003), and DAGchainer (Haas et al., 2004). Both ADHoRe, for Automatic Detection of Homologous Regions, and DiagHunter, do an all vs. all BLASTP search to arrive at homology scores. Both algorithms' hit clustering process,

although different, relies on an iterative process that connects clusters along a diagonal. Also, both ADHoRe and DiagHunter keep track of gene orientation and collapses (ADHoRe) or weights against (DiagHunter) tandem repeats to avoid disrupting the diagonal. DiagHunter also has a compressing parameter that brings genes "closer together." This can be used to adjust for gene density and to speed up alignment. DiagHunter, while not performing a direct sequence alignment, makes up for this weakness with its speed—under a minute on a PC for a comparison of *Arabidopsis* vs. *Arabidopsis*—and it outperforms MUMmer, BLASTZ and PatternHunter at identifying macroscale features. A third algorithm, DAGchainer, also starts with matches of proteins from BLASTP, but uses a directed acyclic graph (DAG) to connect these. In the DAG, the score is related to the strength of the homology, as measured by the BLASTP E-value, as well as the distance between the genes and how tightly they are on the diagonal. Dynamic programming is used to select the highest scoring paths through the DAG until all high scoring paths are exhausted. The process is repeated, using a comparison against the reversed genome to account for both orientations. DAGchainer took only 6 seconds with a Pentium 4 processor to find duplications in the *Arabidopsis* genome, although it appears to only have an accuracy slightly above 90%. These three programs also have the advantage over algorithms like BLASTZ and MUMmer in that they directly give the identity of the syntenic genes, while the other algorithms require an additional processing step to arrive at this.

The last algorithm to be discussed is the UniMarker method (Liao et al., 2004). Unimarker achieves high speed by bypassing direct sequence alignment. They first input the location of all 16-mers that occur only once in the genome, which can be efficiently located (Chen et al., 2002). They then scan the genomes with a 50 kb window that slides at 10 kb intervals, and homologous regions of the two genomes are paired by the number of shared

unique 16-mers within the window. The large scanning window steps avoid small local similarities that could confound results. A number of other processing steps were used to filter results and increase accuracy. This method is very fast; it can map the human and mouse genomes in only one day with a Pentium IV. One drawback of UniMarker is that it cannot compare distantly related genomes due to excessive noise. Unimarker was designed to find orthologous regions of the genome, not to do a fine-grained sequence analysis, but as a result is very fast while still having a high accuracy as verified by comparisons with BLASTZ.

From this discussion, it is clear that there are many approaches to aligning whole genomes and identifying syntey, each one with its own advantages and drawbacks. No single algorithm can simultaneously maximize speed, memory requirements, sensitivity, and specificity. Although the rare algorithm is simultaneously able to make improvements in multiple characteristics, generally, algorithms involve trade-offs. The most important of these is probably an algorithm's computing burden vs. its sensitivity. Likewise, some algorithms are good at aligning microscale features with high sensitivity, while others shine at finding macroscale homology. Some focus on aligning the whole genome in an unbiased way, while other started with annotations and only align coding regions, but do so very efficiently. There is, then, no single best algorithm, but rather, only best algorithms for one's particular use. One approach to overcoming each algorithm's strengths and weaknesses and to increase accuracy is to integrate the data from alignments with different algorithms, although this adds to the computational time needed. Data integration from multiple sources, along with further algorithm advancements and refinements, promises to yield outstanding alignments between the wealth of genomes currently being sequenced.

Works Cited

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. 1990. Basic local alignment search tool. *J. Mol. Biol*. 215, 403-410.

Altschul, S. F., Madden, T. L., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. 1997. Gapped BLAST and PSI-BLAST—a new generation of protein database search programs. *Nucleic Acids Res.* 25, 3389-3402.

Cannon, S. B., Kozik, A., Chan, B., Michelmore, R., and Young, N. D. 2003. DiagHunter and GenoPix2D: programs for genomic comparisons, large-scale homology discovery and visualization. *Genome Biol*. 4, R68.

Chen, L. Y., Lu, S. H., Shih, E. S., and Hwang, M. J. 2002. Single nucleotide polymorphism mapping using genome-wide unique sequences. *Genome Res*. 12, 1106-1111.

Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O., and Salzberg, S. L. 1999. Alignment of whole genomes. *Nucleic Acids Res.* 27, 2369-2376.

Delcher, A. L., Phillippy, A., Carlton, J., and Salzberg, S. L. 2002. Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res.* 30, 2478-2483.

Fleischmann, R.D. et al. 1995. Whole-genome random sequencing and assembly of Haemophilus influenza Rd. *Science* 269, 496-512.

Haas, B. J., Delcher, A. L., Wortman, J. R., and Salzberg, S. L. 2004. DAGchainer: a tool for mining segmental genome duplications and synteny. *Bioinformatics* 20, 3643-3646.

Hunt, E., Hanlon, N., Leader, D. P., Bryce, H., and Dominiczak, A. F. 2004. The visual language of synteny. *OMICS* 8, 289-305.

Kent, W. J. 2002. BLAT—The BLAST-like Alignment Tool. *Genome Res.* 12, 656-664.

Kihara, D., and Kanehisa, M. 2000. Tandem cluster of membrane proteins in complete genome sequences. *Genome Res*. 10, 731-743.

Kurtz, S. 1999. Reducing the space requirement of suffix trees. *Software Pract. Experience* 29, 1149-1171.

Lander, E. S. et al. 2001. Initial sequencing and analysis of the human genome. *Nature* 409, 860-921.

Li, M., Ma, B., Kisman, D., and Tromp, J. 2004. PatternHunter II: highly sensitive and fast homology search. *J Bioinform Comput Biol*. 3, 417-439.

Liao, B. Y., Chang, Y. J., Ho, J. M., and Hwang, M. J. 2004. The UniMarker (UM) method for synteny mapping of large genomes. *Bioinformatics* 20, 3156-3165.

Ma, B., Tromp, J., and Li, M. 2002. PatternHunter: faster and more sensitive homology search. *Bioinformatics* 18, 440-445.

Needleman, S. and Wunsch, C. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol*. 48, 443-453.

Ning, Z., Cox, A. J., and Mullikin, J. C. 2001. SSAHA: A Fast Search Method for Large DNA Databases. *Genome Res.* 11, 1725-1729.

Pearson, W. R. 2000. Flexible sequence similarity searching with the FASTA3 program package. *Methods Mol. Biol*. 132, 185-219.

Pevzner, P., and Tesler, G. 2003. Genome Rearrangements in Mammalian Evolution: Lessons From Human and Mouse Genomes. *Genome Res.* 13, 37-45.

Pop, M., Phillippy, A., Delcher, A. L., and Salzberg, S. L. 2004. Comparative genome assembly. *Brief Bioinform*. 5, 237-248.

Schwartz, S., Kent, W. J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R. C., Haussler, D., and Miller, W. 2003. Human-Mouse Alignments with BLASTZ. *Genome Res*. 13, 103-107.

Schwartz, S., Zhang, Z. Frazer, K. A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R. C., and Miller, W. 2000. PipMaker—a web server for aligning two genomic DNA sequences. *Genome Res*. 10, 577-586.

Smith, T. and Waterman, M. 1981. Identification of common molecular subsequences. *J. Mol. Biol*. 147, 195-197.

Tesler, G. 2002. GRIMM: genome rearrangements web server. *Bioinformatics* 18, 492-493.

Tomii, K., and Kanehisa, M. 1998. A comparative analysis of ABC transporters in complete microbial genomes. *Genome Res*. 8, 1048-1059.

Vandepoele, K., Saeys, Y., Simillion, C., Raes, J., and Van de Peer, Y. 2002. The Automatic Detection of Homologous Regions ADHoRe and Its application to Microcolinearity between *Arabidopsis* and Rice. *Genome Res*. 12, 1792-1801.

Venter, J.C. et al. 2001. The sequence of the human genome. *Science* 291, 1304-1351.