

Phylogenies from Gene Order Data: A Detailed Study of Breakpoint Analysis

Bernard M.E. Moret
Dept. of Computer Science
University of New Mexico
Albuquerque, NM 87131, USA

Stacia Wyman
Dept. of Computer Science
University of Texas
Austin, TX 78712

David A. Bader
Dept. of Electrical & Computer Engineering
University of New Mexico
Albuquerque, NM 87131, USA

Tandy Warnow
Dept. of Computer Science
University of Texas
Austin, TX 78712

Mi Yan
Dept. of Electrical & Computer Engineering
University of New Mexico
Albuquerque, NM 87131

Gene order data and phylogenies derived from them may prove crucial in answering some fundamental open questions in biomolecular evolution. Yet there are very few techniques available for such phylogenetic reconstructions. One method is *breakpoint analysis*, developed by Blanchette and Sankoff¹ for reconstructing the “breakpoint phylogeny.” Our earlier studies^{6,7} confirmed the usefulness of the breakpoint phylogeny approach, but also found that *BPANALYSIS*, the implementation of breakpoint analysis developed by Blanchette and Sankoff, was too slow to use on all but very small datasets. In this paper, we report on a new implementation of breakpoint analysis. Our faster (by two orders of magnitude) and flexible implementation allowed us to conduct studies on the characteristics of breakpoint analysis, in terms of running time, quality, and robustness, as well as to analyze datasets that had been considered out of reach of such approaches until now. We report on these findings and also discuss future steps that our new implementation will enable us to take.

1 Introduction

Some organisms have a single chromosome or contain single-chromosome organelles (such as mitochondria or chloroplasts), the evolution of which is largely independent of the evolution of the nuclear genome. Many single-chromosome organisms and organelles have circular chromosomes. Given a particular strand from a single chromosome, whether linear or circular, we can infer the ordering of the genes, along with directionality of the genes, thus representing each chromosome by an ordering (linear or circular) of signed genes. (Note that picking the complementary strand produces a different ordering, in which the genes appear in the reverse direction and reverse order.) The evolutionary process that operates on the chromosome can thus be seen as a transformation of signed orderings of genes.

Gene orderings represent a new source of data for phylogeny reconstruction. Because evolutionary changes on gene orders operate at a much higher level than nucleotide or amino-acid changes on sequence data, they may be more realistic and

more likely to represent survivable mutations. Thus many biologists have embraced gene order data in their phylogenetic work.^{19,20,22}

The first heuristic for reconstructing phylogenetic trees from gene order data was introduced by Blanchette *et al.*¹. This technique, called *breakpoint analysis* and implemented by its authors in the code `BPAnalysis`²³, was used to infer phylogenies for a variety of small datasets^{2,24}. Cosner *et al.*^{6,7} studied this technique (and one they introduced) experimentally, using simulations of genome evolution; they determined that `BPAnalysis` was quite accurate, but slow enough that it could only be applied to problems with small numbers of genomes containing a small number of genes.

Breakpoint analysis as described by Blanchette *et al.* is an exhaustive search through tree space, with a point estimation heuristic. It can be decomposed in three levels: at the highest level, it generates all tree topologies; at the middle level, each tree is given internal node labels through a heuristic iterative procedure that repeatedly finds the median of three neighboring labels; and at the lowest level, the median of three genomes is computed exactly through a reduction to the *Travelling Salesperson Problem (TSP)*. Because the TSP is NP-hard, exact solutions are rarely feasible for large problems, but the instances generated by the reduction are of medium size and possess a very rigid structure that makes them easier to solve. Nevertheless, optimal solutions become unattainable for large genomes (hundreds of genes). Fortunately, the TSP is the best studied of all optimization problems¹⁰; numerous approximation algorithms have been designed for it, including the *Lin-Kernighan (LK)* algorithm¹², the epitome of a successful heuristic for an NP-hard problem.

Our implementation of breakpoint analysis allows the use of any TSP heuristic in lieu of the exact solver. We also implemented an exact solver along the lines proposed by Sankoff and Blanchette²³ to test the limits of the exact approach. Through careful design and a process that we helped pioneer and that we termed *algorithmic engineering*^{16,17}, we produced a software suite that, when run with the exact solver, is almost two orders of magnitude faster than `BPAnalysis`. This tool enabled us to study breakpoint analysis in some detail, in terms both of quality of solutions and running time, as well as to prepare a parallel implementation that enables us to analyze significant biological datasets within a reasonable amount of time.

2 Distances between Genomes Based on Gene Orders

We assume a fixed set of genes $\{g_1, g_2, \dots, g_n\}$. Each genome is then an ordering (circular or linear) of some multisubset of these genes, each gene given with an orientation that is either positive (g_i) or negative ($-g_i$). A linear genome is simply a permutation on this multisubset, while a circular genome can be represented in the same way under the implicit assumption that the permutation closes back on itself. In tracing the evolutionary history of a collection of single-chromosome genomes, we currently only consider inversions, transpositions, and transversions (inverted transpositions),

because these events only rearrange gene orders (no repeats, insertions or deletions).

Let G be the genome with signed ordering (linear or circular) g_1, g_2, \dots, g_n . An *inversion* between indices i and j , for $i < j$, produces the genome with linear ordering

$$g_1, g_2, \dots, g_{i-1}, -g_j, -g_{j-1}, \dots, -g_i, g_{j+1}, \dots, g_n$$

If we have $j < i$, we can still apply an inversion to a circular (but not linear) genome by simply rotating the circular ordering until the two indices are in the proper relationship. The *inversion distance* between two genomes is the minimum number of inversions needed to transform one genome into another. The inversion distance between two genomes is computable in polynomial time for signed genomes^{8,11}, but is NP-hard³ in the unsigned case.

Another distance between genomes to consider, which is not directly an evolutionary metric, is the *breakpoint distance*. Given two genomes G and G' on the same set of genes, a breakpoint in G is defined as an ordered pair of genes (g_i, g_j) such that g_i and g_j appear consecutively in that order in G , but neither (g_i, g_j) nor $(-g_j, -g_i)$ appear consecutively in that order in G' . For instance, if $G = g_1, g_2, -g_4, -g_3$ and $G' = g_1, g_2, g_3, g_4$, then there are exactly two breakpoints in G : $(g_2, -g_4)$, and $(-g_3, g_1)$. The pair $(-g_4, -g_3)$ is not a breakpoint in G' since (g_3, g_4) appear consecutively and in that order in G' . The *breakpoint distance* is the number of breakpoints in G relative to G' (or vice-versa, since the measure is symmetric). It has long been known that the breakpoint distance is at most twice the inversion distance for any two genomes. For some datasets, however, there can be a close-to-linear relationship between the breakpoint distance and the inversion distance, in which case the breakpoint distance can serve as a surrogate for the more biologically significant, but more computationally demanding, inversion distance.

3 Phylogeny Reconstruction from Gene Order Data

Maximum Parsimony for Rearranged Genomes (MPRG): Assume that we are given a tree in which each node is labelled by a genome. We define the cost of the tree to be the sum of the costs of its edges, where the cost of an edge is the edit distance between the two genomes that label the endpoints of the edge. Finding the tree of minimum cost for a given set of genomes and a given definition of the edit distance is the problem of *Maximum Parsimony for Rearranged Genomes (MPRG)*; the optimal trees are called the maximum-parsimony trees. MPRG seems to be the optimization criterion of choice; indeed, most approaches to the reconstruction of phylogenetic trees from gene order data have explicitly sought to find the maximum-parsimony tree with respect to some definition of genomic distances (inversion distances or a weighted sum of inversions, transpositions, and transversions). However, all these problems are NP-hard or of unknown computational complexity. Even the fundamental problem of computing optimal labels (genomes) for the internal nodes is very difficult: when only inversions are allowed, it is NP-hard even when there are only three leaves⁴.

Breakpoint Phylogeny Blanchette *et al.*² recently proposed a new optimization problem for phylogeny reconstruction on gene order data. In this problem, the tree sought is that with the minimum number of breakpoints rather than that with the minimum number of evolutionary events. When a close-to-linear relationship exists between breakpoint and inversion distances, the tree with the minimum number of breakpoints may be assumed to be close to optimal with respect to the number of evolutionary events. Indeed, some of our earlier experiments⁶ strongly supported this assumption and showed that, when the rates of evolution are low relative to the number of genes in the genomes, the breakpoint phylogeny is very close to the true tree. Thus, reconstructing the breakpoint phylogeny is a reasonable approach to phylogeny reconstruction under these conditions.

BPAnalysis The first method for reconstructing the breakpoint phylogeny was developed by Sankoff and Blanchette²³, who were also the first to propose the breakpoint phylogeny; their method, called `BPAnalysis`, combines exhaustive search, heuristics, and exact optimization. Computing the breakpoint phylogeny is NP-hard for the case of just three linear signed genomes²¹, a special case known as the *Median Problem for Breakpoints (MPB)*. Blanchette *et al.* showed that the MPB reduces to the Travelling Salesman Problem (TSP)¹⁰ and designed special heuristics for the resulting instances of TSP. Their overall heuristic for the breakpoint phylogeny thus has the following structure:

```
For each tree topology in turn, do:
    Relabel internal nodes with gene orders by computing medians
    of triplets of genomes iteratively (until no change occurs),
    using the TSP reduction; score the resulting tree.
Return the tree with the minimum number of breakpoints.
```

This technique is computationally intensive on several levels. First, the number of unrooted binary trees on n leaves is exponential in n (specifically it is $(2n - 5)!! = (2n - 5) \cdot (2n - 7) \cdot \dots \cdot 3$), so that the outer loop is exponential in the number of genomes. Secondly, the inner loop itself is computationally intensive, since computing the median of three genomes is NP-hard²¹ and because the technique used by Blanchette *et al.* involves solving many instances of TSP in a reduction where the number of cities is twice the number of genes in the input. Finally, the number of instances of TSP can be quite large, since the procedure iterates until no further change of labeling occurs within the tree. Thus, the computational complexity of the entire algorithm is exponential in *each* of the number of genomes and the number of genes.

4 Study Objectives

In earlier studies^{6,7}, we attempted to compare `BPAnalysis` to a method we developed called *Maximum Parsimony on Binary Encodings (MPBE)*, which also attempts to reconstruct the breakpoint phylogeny. Our attempt was thwarted because

BPAnalysis was much too slow. Our dataset (a collection of Campanulaceae⁶) had 13 genomes of 105 gene segments. On this set, MPBE completed its search in less than a second, but we estimated that BPAnalysis would take well over 200 years to complete—an estimate based on the average number of trees processed by the code per unit time (120 trees a minute on our small SGI platform) and extended to the 13,749,310,575 tree topologies on 13 leaves. Blanchette *et al.* did complete their analysis of the metazoan dataset, which has 11 genomes on a set of 37 genes. This is a much easier problem, as there are far fewer trees to examine (only 2,027,025) and as scoring each tree involves solving a smaller number of TSP instances on a much smaller number of cities (74 rather than 210). Overall, it is clear that datasets of sizes such as ours are currently too large to be fully analyzed by BPAnalysis.

Our objective was therefore to develop a much faster implementation of BPAnalysis, to allow the substitution of the exact TSP solver by an approximation method, and to investigate the use of massive parallelism in searching through tree space, all in order to make breakpoint analysis possible for up to 12–14 genomes with hundreds of genes.

5 Our Implementation

Exploring tree space requires the efficient generation of tree topologies; because we also intend to use a branch-and-bound technique in tree space to enable us to analyze datasets with more than 12–14 genomes, we need a generating mechanism that is interruptible and restartable at any point. We chose to generate a preorder encoding of the tree, then to produce the topology from the encoding; this also enables us to produce only trees that are refinements of a given constraint tree—a capability that was also built into BPAnalysis.

Labeling the internal nodes of a tree is the most challenging part of the problem—indeed, no algorithm is known that would produce an optimal solution for more than 3 leaves. As observed, the problem is NP-hard even for a 3-leaf tree—but for 3 leaves it is possible to produce an optimal solution. Thus we followed the strategy of BPAnalysis: do a postorder traversal of the tree; at each internal node, use the labels of its three neighbors to define an instance of the MPB, solve that problem, and assign the new label to the node if the number of breakpoints is thereby lowered; and repeat the entire process until no change occurs. This process is enormously expensive and quite wasteful—an NP-hard problem must be solved at each internal tree node, over and over, with most solutions discarded because they do not bring about any improvement. Our implementation avoids much of this unnecessary expense by only generating an MPB problem for nodes that saw at least one of their three neighbors relabeled over the last pass, but retains a postorder tree traversal as an organizing principle.

Each MPB problem is solved through reduction to a TSP instance; the instance produced has 2 cities for each gene in the genome. The number of such instances

solved in the analysis of a dataset is very large—and, of course, the TSP problem is itself NP-hard. Therefore we spent most of our design effort on the TSP solver. We used the *Concorde* library⁵ for two of our three solvers—the chained and the simple versions of the famous Lin-Kernighan heuristic¹². These are approximation algorithms that run in (fairly) low polynomial time and return excellent approximations—typically within a few percent of optimal for the simple version and even closer for the slower chained version. (The chained version restarts the basic one after it has finished, feeding it a modified version of the previous solution, until some large number of consecutive restarts have failed to produce any improvement—thus its running time is a significant multiple of the running time of the simple version.) We also implemented a simple include-exclude depth-first search with pruning, along the lines of the `BPAnalysis` code, but with more streamlined data structures, better bounding (we use the same bounding function, but more information of the eligibility of each edge), and, most significantly, some features tailored to the very special nature of the instances generated in the reduction. For instance, of the $\Theta(n^2)$ edges of an instance, only $\Theta(n)$ of them are non-trivial (those that correspond to adjacent genes segments in the three genomes); our implementation only considers nontrivial edges, treating the others as an undifferentiated pool—a refinement that allows each step in the search to run in linear rather than quadratic time.

Additionally, we built a parallel implementation, based on MPI, in which each processor generates its own batch of trees, analyzes them, collects the best, then confers with the other processors to produce the overall best tree. This type of “embarrassing” parallelism works well on clusters of machines and thus allows the use of massive parallelism to solve large datasets.

6 Experimental Procedure

We had several objectives for our experiments. First we wanted to compare the raw running times of our three versions and of `BPAnalysis` and investigate their dependency on the number of genomes, the number of genes, and the rate of evolution in the model. Secondly, we wanted to understand the source of the various costs of our procedures to give us the understanding we would need in developing further versions. Thirdly, we wanted to study the impact on the quality of solutions of using an approximation algorithm for TSP in lieu of an exact one. Finally, with a fast implementation of breakpoint analysis, we wanted to compare the quality of its solutions to that of the solutions produced by our (very much faster) MPBE approach.

The standard approach to the assessment of quality in phylogeny reconstruction is simulation⁹. In such a simulation, one picks a model of evolution, then one generates a tree topology and uses the model to generate a collection of genomes on the tree; that collection is then fed to the reconstruction algorithm and the output compared to the model tree. Our focus is more on running time than quality of reconstruction—and

we can use the exact version of breakpoint analysis for comparison with the quality of solutions returned by our approximate versions, so we designed experiments with and without the full simulation process.

In order to obtain statistically significant data, we followed the recommendations of McGeoch^{14,15} and Moret¹⁷. We used *runs* of 10 *trials* each—each trial is one dataset—and retained only the average value from each run; we then examined the values derived from a number of independent runs for consistency. This method produces robust results even when, as in our case, the enormous size of the sample space precludes any fair sampling.

7 Experiments

In order to test our TSP solvers, we ran a number of tests with just 3 genomes. These problems have only one tree topology and only one internal node to label, so that only one call is made to the TSP solver. Because we expect the exact methods to start failing as the number of genomes grows large or as the rate of evolution increases, we used 6 different numbers of genes per genome and up to three rates of evolution. We generated the 3-genome problems under the Nadeau-Taylor model of evolution¹⁸ and used runs of 10 trials each. In order to assess the ability of our codes to solve large problems, we ran all four programs on our Campanulaceae dataset (13 genomes of 105 genes)—not to completion, but long enough to obtain an accurate count of the number of trees processed per unit time and thus be able to predict the running time to completion. We repeated this experiment on other datasets in order to get a sense of scaling with the number of genes and the number of genomes. We then used the same datasets to study the behavior of the tree (re)labeling procedure. Our experiments were run on a 550MHz Pentium-III laptop with 128MB of memory running Linux.

8 Results and Discussion

Experiments on 3-genome problems Figure 1 shows the running times of our three versions and of `BPAnalysis` on 3-genome problems of increasing sizes and at different rates of evolution. The rates $r = 2, 4, 8$ reflect the number of (randomly chosen) inversions on each tree edge. The datasets of 80 genes with $r = 4$ and the datasets with $r = 8$ in general caused erratic behavior with all the solvers, but most obviously with the two exact solvers; for instance, `BPAnalysis` could not solve within 15 mins of computation more than 10% of the problems.

Figure 2 presents the same data, this time for each evolutionary rate so as to facilitate comparisons of running times. We know from algorithm analysis that the two LK solvers have quadratic to cubic running times and the exact solvers have exponential running times. Our figures clearly demonstrate the exponential behavior of `BPAnalysis`, but our exact solver stays in a flat part of its exponential curve all the way to 320 genes; the two LK solvers show at least quadratic behavior. With

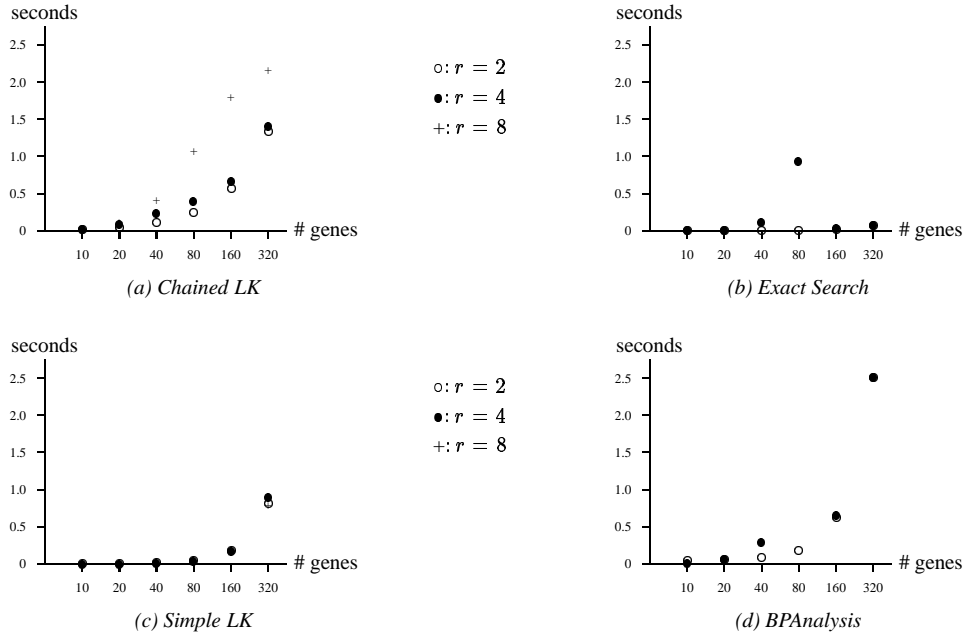


Figure 1: Speed of the various codes on 3-genome problems of various sizes under 3 different rates of evolution

larger evolutionary rates, the exact solvers suffer—indeed, the problems rapidly become intractable—whereas the Chained LK solver slows down a bit and the simple LK solver not at all. Because high rates of evolution induce numerous break points, the instances of the TSP created under high rates of evolution have relatively undiffer-

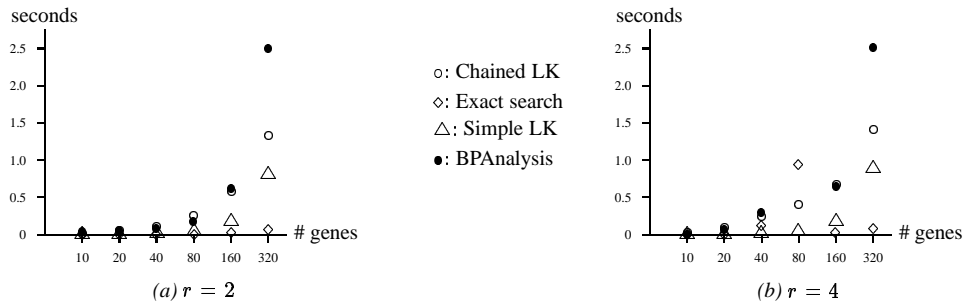


Figure 2: Relative running times of the four methods on 3-genome problems of various sizes

entiated edges costs—most of the edges have maximal or near maximal cost, making it very difficult to find an optimal solution quickly.

Quality of approximation of LK heuristics Figure 3 shows the percentage by which the solutions returned by the simple LK solver exceed the optimal. In con-

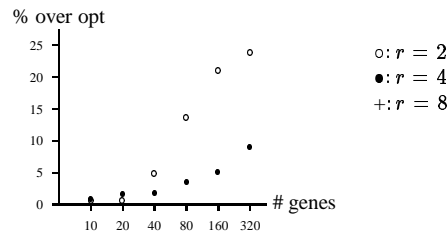


Figure 3: Percentage excess over optimal for LK solutions

trast, note that the Chained LK solver returned optimal solutions for all of these test instances. The percentage error is larger for smaller evolutionary rates because, for these lower rates, the number of breakpoints in the optimal solution is quite small, thus magnifying percentages. The error grows with the number of cities at a fairly steep rate; while it is known that the LK solver tends to have increasingly large error as the input size grows,¹⁰ the growth shown in Figure 3 is much larger than what has been reported. We believe that this is due to the very limited range of edge lengths in the TSP instance (three values only), which tends to mask the potential effect of small changes to an existing tour.

Tree processing rates In order to estimate how long each method would take to process a sizable data set, we ran each on 3 different datasets (the Campanulaceae set and two simulated datasets, one with a large number of genes and one with a sizable number of taxa) and let it process several hundred to several thousand trees until a fixed time bound was attained. We then normalized the result to obtain an estimated time required by each method to process 1,000 trees. Table 1 shows the results; in the table, $n/N/r$ denotes a problem with n genomes, N genes, and r inversions per model tree edge. Tree rates for the 5/200/2 problem are for all 15 trees of the problem, not for 1,000 trees. The very high processing rate of our exact solver (from 30 to 100 times faster than `BPAnalysis` makes it possible to solve problems with 10–12 genomes on a single processor; with our parallel implementation and with the use of a supercluster, problems of 13–14 genomes can now be handled. Chained LK is much too slow to be of use and even simple LK, while often faster than `BPAnalysis`, is far slower than our exact solver.

Table 1: Tree processing rates for all 4 methods on 4 datasets

method	13/100/-	8/50/2	10/20/-	5/200/2
Chained LK	90m	2700s	25m	70s
Simple LK	33m	290s	125s	21s
Exact solver	90s	14s	8s	1s
BPAAnalysis	50m	570s	9m	44s
# trees	1000	1000	1000	15

Labeling details Having optimized the TSP solvers as much as we could, we turned our attention to the question of how often they were called and how often they proved useful. BPAAnalysis calls the solver at each internal node in each pass of the relabeling algorithm; our exact solver saves a large amount of work by not calling a TSP solver when none of the genomes to be used in the computation has changed, yet may still make a large number of unproductive calls. Thus we examined how many relabeling passes (tree traversals) took place and how many nodes were actually relabeled (as opposed to the number for which a new label was generated, but not necessarily used). Figure 4 shows the distribution of relabelings during tree traversals for our exact solver and for the simple LK solver, for two different trees. The exponential

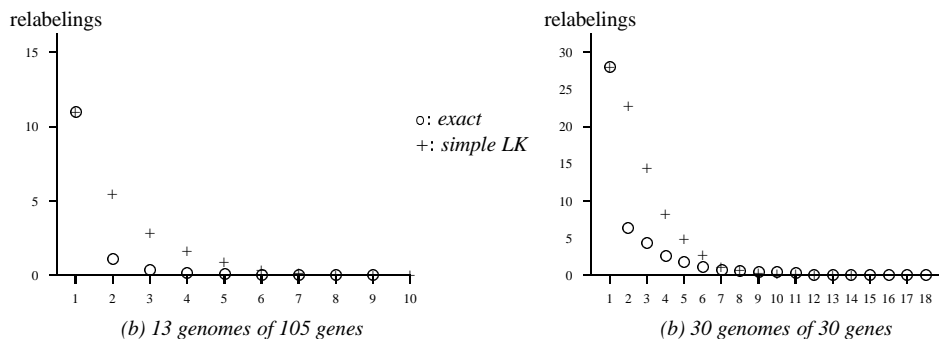


Figure 4: Average number of nodes relabelled at each pass

decay is evident. Interestingly, the decay is much faster with the exact solver than with the simple LK solver, showing that accurate solutions to TSP instances make a global difference in the behavior of the algorithm.

The parallel implementation Our parallel implementation is very predictable—we simply used it to verify that a cluster of p processors would provide us with a linear speed-up very close to a factor of p . Thus, using our fast exact solver, we can now process the 13-genome, 105-gene Campanulaceae dataset—the same dataset that would

have required over 200 years of processing with `BPAnalysis` on an SGI machine—in well less than a day on the Alliance supercluster *Los Lobos*,¹³ a 512-processor Pentium cluster at the University of New Mexico.

9 Conclusions and Future Work

We have presented a new implementation of breakpoint analysis, one that improves on the original `BPAnalysis` by nearly two orders of magnitude—an improvement reached through the application of algorithmic engineering. Our new implementation makes it possible to analyze significantly larger datasets; combined with massive parallelism, it reduces the running time of an analysis of a set of Campanulaceae from two centuries down to less than a day. We discussed various aspects of the performance of our implementation and the original `BPAnalysis`.

Our implementation uses a random initialization of labels, which has the virtue of simplicity, but may not allow the relabeling heuristic to discover solutions as good as it would with a more sophisticated initialization. This is a topic that was addressed by Blanchette *et al.*, who eventually chose a more complex, but better-informed initialization. Breakpoint scores may not be the measure of choice; since we can compute the inversion score efficiently¹¹, it may be beneficial to replace the tree scoring routine by one that counts the number of inversions.

These and other improvements and additions pale against the main drawback of the approach: the enumeration of all tree topologies is simply impossible for 20 or more genomes. Thus, if breakpoint analysis through relabeling is to succeed on larger problems, an implicit exploration of tree space is necessary.

References

1. Blanchette, M., Bourque, G., & Sankoff, D., “Breakpoint phylogenies,” in *Genome Informatics 1997*, Miyano, S., and Takagi, T., eds., Univ. Academy Press, Tokyo, 25–34.
2. Blanchette, M., Kunisawa, T., & Sankoff, D., “Gene order breakpoint evidence in animal mitochondrial phylogeny,” *J. Mol. Evol.* **49** (1999), 193–203.
3. Caprara, A., “Sorting by reversals is difficult,” *Proc. 1st Conf. Computational Molecular Biology RECOMB97*, ACM Press, New York (1997), 75–83.
4. Caprara, A., “Formulations and hardness of multiple sorting by reversals,” *Proc. 3rd Conf. Computational Molecular Biology RECOMB99*, ACM Press, New York (1999), 84–93.
5. Applegate, D., Bixby, R., Chvátal, V., & Cook, W., “CONCORDE: Combinatorial Optimization and Networked Combinatorial Optimization Research and Development Environment,” available at www.keck.caam.rice.edu/concorde.html
6. Cosner, M.E., Jansen, R.K., Moret, B.M.E., Raubeson, L.A., Wang, L.-S., Warnow, T., & Wyman, S., “A new fast heuristic for computing the breakpoint phylogeny and experimental phylogenetic analyses of real and synthetic data,” *Proc. 8th Int’l Conf. on Intelligent Systems for Molecular Biology ISMB-2000*, to appear.
7. Cosner, M.E., Jansen, R.K., Moret, B.M.E., Raubeson, L.A., Wang, L.S., Warnow, T.,

- & Wyman, S., "An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae," *Proc. Gene Order Dynamics, Comparative Maps, and Multigene Families DCAF-2000*, Montreal (2000).
8. Hannenhalli, S., & Pevzner, P.A., "Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals)," *Proc. 27th Ann. ACM Symp. on Theory of Computing*, ACM Press (1995), 178–189.
 9. Hillis, D.M., "Approaches for assessing phylogenetic accuracy," *Syst. Biol.* **44** (1995), 3–16.
 10. Johnson, D.S., & McGeoch, L.A., "The traveling salesman problem: a case study," in *Local Search in Combinatorial Optimization*, E. Aarts & J.K. Lenstra, eds., John Wiley, New York (1997), 215–310.
 11. Kaplan, H., Shamir, R., & Tarjan, R.E., "Faster and simpler algorithm for sorting signed permutations by reversals," *Proc. 8th ACM-SIAM Symp. on Discrete Algorithms SODA97*, ACM Press (1997), 344–351.
 12. Lin, S., & Kernighan, B.W., "An effective heuristic algorithm for the traveling salesman problem," *Operations Res.* **21** (1973), 498–516.
 13. Los Lobos, at www.ahpcc.unm.edu/Hardware/LosLobos.html.
 14. McGeoch, C.C., "Analyzing algorithms by simulation: variance reduction techniques and simulation speedups," *ACM Comput. Surveys* **24**, 2 (1992), 195–212.
 15. McGeoch, C.C., "Toward an experimental method for algorithm simulation," *INFORMS J. Comput.* **8** (1996), 1–15.
 16. McGeoch, C.C., & Moret, B.M.E., "How to present a paper on experimental work with algorithms," *SIGACT News* **30**, 4 (1999), 85–90.
 17. Moret, B.M.E., "Towards a discipline of experimental algorithmics," to appear in *Proc. 5th DIMACS Challenge*, available at www.cs.unm.edu/~moret/dimacs.ps.
 18. Nadeau, J.H., & Taylor, B.A., "Lengths of chromosome segments conserved since divergence of man and mouse," *Proc. Nat'l Acad. Sci. USA* **81** (1984), 814–818.
 19. Olmstead, R.G., & Palmer, J.D., "Chloroplast DNA systematics: a review of methods and data analysis," *Amer. J. Bot.* **81** (1994), 1205–1224.
 20. Palmer, J.D., "Chloroplast and mitochondrial genome evolution in land plants," in *Cell Organelles*, Herrmann, R., ed., Springer Verlag (1992), 99–133.
 21. Pe'er, I., & Shamir, R., "The median problems for breakpoints are NP-complete," *Elec. Colloq. on Comput. Complexity*, Report 71, 1998.
 22. Raubeson, L.A., & Jansen, R.K., "Chloroplast DNA evidence on the ancient evolutionary split in vascular land plants," *Science* **255** (1992), 1697–1699.
 23. Sankoff, D., & Blanchette, M., "Multiple genome rearrangement and breakpoint phylogeny," *J. Computational Biology* **5** (1998), 555–570.
 24. Sankoff, D., Leduc, G., Antoine, N., Paquin, B., Lang, B.F., & Cedergren, R., "Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome," *Evolution* **89** (1992), 6575–6579.