

# Industrial Applications of High-Performance Computing for Phylogeny Reconstruction

David A. Bader<sup>a</sup>, Bernard M.E. Moret<sup>b</sup>, and Lisa Vawter<sup>c</sup>

<sup>a</sup>Electrical and Computer Engineering Department,  
University of New Mexico, Albuquerque, NM 87131 USA

<sup>b</sup>Computer Science Department, University of New Mexico, Albuquerque, NM 87131 USA

<sup>c</sup>Bioinformatics UW2230, PO Box 1539, Glaxo SmithKline, King of Prussia, PA 19406 USA

## ABSTRACT

Phylogenies (that is, tree-of-life relationships) derived from gene order data may prove crucial in answering some fundamental open questions in biomolecular evolution. Real-world interest is strong in determining these relationships. For example, pharmaceutical companies may use phylogeny reconstruction in drug discovery for discovering synthetic pathways unique to organisms that they wish to target. Health organizations study the phylogenies of organisms such as HIV in order to understand their epidemiologies and to aid in predicting the behaviors of future outbreaks. And governments are interested in aiding the production of such foodstuffs as rice, wheat and potatoes via genetics through understanding of the phylogenetic distribution of genetic variation in wild populations. Yet few techniques are available for difficult phylogenetic reconstruction problems. Appropriate tools for analysis of such data may aid in resolving some of the phylogenetic problems that have been analyzed without much resolution for decades.

With the rapid accumulation of whole genome sequences for a wide diversity of taxa, especially microbial taxa, phylogenetic reconstruction based on changes in gene order and gene content is showing promise, particularly for resolving deep (i.e., ancient) branch splits. However, reconstruction from gene-order data is even more computationally expensive than reconstruction from sequence data, particularly in groups with large numbers of genes and highly-rearranged genomes. We have developed a software suite, GRAPPA, that extends the breakpoint analysis (BPAnalysis) method of Sankoff and Blanchette while running much faster: in a recent analysis of chloroplast genome data for species of Campanulaceae on a 512-processor Linux supercluster with Myrinet, we achieved a one-million-fold speedup over BPAnalysis. GRAPPA can use either breakpoint or inversion distance (computed exactly) for its computation and runs on single-processor machines as well as parallel and high-performance computers.

**Keywords:** high-performance computing, computational genomics, phylogeny reconstruction, breakpoint analysis, gene rearrangement, drug discovery

## 1. INTRODUCTION

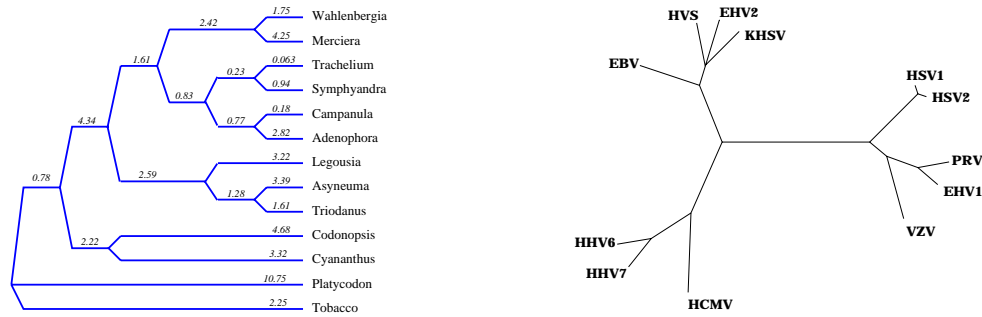
Curiosity about pattern and process in the evolution of diversity has motivated many biologists and natural historians to study phylogenetics. Phylogenetic analysis attempts to reconstruct, from data generated from extant species, the evolutionary history of the group under study. The history is generally represented by a bifurcating (binary) tree, a phylogenetic tree. Such phylogenies, constructed from molecular and genetic data with the help of computers, are proving to be essential elements of the bioinformatics toolkit used by the pharmaceutical industry.

In this paper, we briefly introduce phylogenetic analysis, surveying some of the major reconstruction methods and the data that they use. Then we list prominent industrial uses of phylogeny reconstruction, most of which necessitate significant computing efforts. We then present some of our own work in reconstructing phylogenies from gene-order data—work that resulted in the widely-used software suite GRAPPA—and discuss the high-performance aspects of our design and implementation.

The organization of this paper is as follows: The following section introduces phylogenetic analysis; Section 3 surveys existing commercial applications of phylogeny reconstruction, and Section 4 briefly reviews the principal computational methods

---

Correspondence: D. A. Bader: e-mail: [dbader@eece.unm.edu](mailto:dbader@eece.unm.edu), telephone: +1.505.277.6724, URL: <http://hpc.eece.unm.edu/>  
B.M.E. Moret: e-mail: [moret@cs.unm.edu](mailto:moret@cs.unm.edu), telephone: +1.505.277.5699, URL: <http://www.cs.unm.edu/~moret>  
L. Vawter: e-mail: [lisa\\_vawter@sbphrd.com](mailto:lisa_vawter@sbphrd.com), telephone: +1.610.270.6357



**Figure 1.** Two phylogenies: some plants of the Campanulaceae (left) and some herpesviruses affecting humans (right)

used for phylogenetic reconstruction. Section 5 gives an overview of algorithm engineering, an emerging discipline that addressed methodologies by which modern algorithm designs can be transformed into efficient, robust code. Section 6 illustrates this approach as applied by members of our group to an existing strategy for from gene-order data. Here, we start with the method of breakpoint analysis<sup>1</sup> and produce the software suite GRAPPA (see, e.g.<sup>2</sup>), which runs at least three orders of magnitude faster than the existing implementation and parallelizes extremely well.<sup>3</sup> In Section 7, we conclude with our personal views about the current impact of high-performance computing applied to discrete optimization problems in computational biology.

## 2. PHYLOGENIES

A phylogeny is an attempt to reconstruct the evolutionary history of a collection of organisms or a family of genes (known as taxa). The basic assumption of process necessary for phylogenetic reconstruction is repeated divergence from ancestral populations or genes. A phylogenetic reconstruction is usually depicted as an evolutionary tree, in which modern taxa are depicted at the leaves and ancestral taxa occupy internal nodes, with the edges of the tree denoting genetic, historical, ancestor-descendant relationships among the taxa. Figure 1 shows two proposed phylogenetic trees, one for several species of Campanulaceae (the bluebell family) and the other for herpesviruses, including the Epstein-Barr virus and the herpesvirus that causes Kaposi's sarcoma. In the left-hand figure, estimates of percent genetic divergence (evolutionary distance) are used to label each edge; in the right-hand figure, the length of each edge is scaled to represent that distance. Neither of these phylogenetic trees infers of the possible characteristics of ancestral taxa at the internal nodes, although some phylogenetic reconstructions (as we shall see) infer such characteristics.

Reconstructing phylogenies is a major component of modern research programs in many areas of biology and medicine (as well as linguistics). Naturally, scientists are interested in phylogenies for the sake of knowledge, but such analyses also have many applied uses. An understanding of evolutionary mechanisms and relationships underlies modern drug discovery, helps researchers understand the defenses of rapidly-mutating viruses such as HIV, is the basis of the design of genetically-enhanced organisms, etc. In developing such an understanding, the reconstruction of phylogenies is a crucial tool, as it allows one to test new models of change.

## 3. COMMERCIAL ASPECTS OF PHYLOGENY RECONSTRUCTION

Simple identification via phylogenetic classification of organisms has, to date, yielded more patent filings than any other use of phylogeny in industry. Just as a public health entomologist might keep a reference collection of mosquitoes or ticks to aid in identification, so do many industry bioinformaticians keep collections of gene sequences. When a microorganism of unknown origin is discovered, the final identification is generally done via sequencing. If the gene sequence from the organism of interest does not match a sequence already in the collection, inference of close relatives to that organism is done via phylogeny. Sequence motifs unique to that organism or to a specific phylogenetic group are often then patented as a means of identification for that organism or group. Examples include sequence motifs used for identification of close relatives of *Mycobacterium tuberculosis* in sputum culture and for differentiating among tuberculosis strains (GenProbe 1991, 1992).<sup>4,5</sup>

A more unusual application of phylogenetic analysis to a practical problem is its use in studying the dynamics of microbial communities. Engelen *et al.* (1998)<sup>6</sup> sequenced genes to identify and quantify microbes in soil before and after pesticide exposure. Because many microbes in many such population studies are novel, their gene sequences are studied phylogenetically in order to understand the composition of the community throughout the experiment.

Phylogenetic analysis has been used in vaccine development. When a vaccine is developed, it is often specific to a particular variant of a cell wall or protein coat component. Halbur *et al.* (1994)<sup>7</sup> used phylogenetic analysis to infer that porcine reproductive and respiratory syndrome virus isolates from the US and Europe were from separate populations, and thus engineered their vaccine to confer immunity against both populations. In the continuing effort to develop HIV vaccines, the population dynamics of HIV are studied by examining DNA markers which originate from disparate populations; initial population boundaries are then established with phylogenetic analysis (GenProbe, 1993).<sup>8</sup>

The phylogenetic distribution of biochemical pathways (Overbeek *et al.*, 2000)<sup>9</sup> is studied in the development of antibacterials and herbicides. Glyphosate (better known as Roundup<sup>TM</sup>, Rodeo<sup>TM</sup> and Pondmaster<sup>TM</sup>) was the first herbicide specifically targeted at a pathway, the shikimate pathway, because that pathway is not present in mammals (Grossbard and Atkins, 1985).<sup>10</sup> Because plants and some microbes rely on this pathway, they are killed by glyphosate. Antimicrobials targeting the shikimate pathway are also being developed (Roberts *et al.*, 1998).<sup>11</sup> In the pharmaceutical industry, the phylogenetic distribution of a pathway is often studied before a drug is developed in order to understand the effective range of an antimicrobial targeted at that pathway<sup>12</sup> (Brown and Warren, 1998). The mevalonate pathway was identified by Wilding *et al.* (2000)<sup>13</sup> as a target for drug development for gram positive cocci-specific antimicrobials. These bacteria contain genes from the eukaryotic mevalonate pathway that were captured by the ancestors to this group of bacteria millions of years ago. The microbial genes have diverged to such an extent that this pathway can be targeted separately in these microbes from our own mevalonate pathway.

Finally, phylogenetic analysis is used in the pharmaceutical industry for predicting the natural ligands for cell surface receptors which are potential drug targets. Just as the speciation process produces hierarchical lineages from a single species, so can a process of duplication and divergence produce hierarchically related gene families from a single ancestral gene. Several large gene families contain drug receptors. In fact, a single family, the G protein coupled receptors (GPCRs) contains more than 40% of the targets of most pharmaceutical companies. Neuromedin U, a potent neuropeptide that causes contraction of smooth muscle, was (correctly) predicted phylogenetically to be a possible ligand for FM3 an orphan GPCR (Szekeres *et al.*, 2000).<sup>14</sup> Chambers *et al.* (2000)<sup>15</sup> used phylogenies to infer a class of modified nucleotides as ligands for a small group of GPCRs. UDP-glucose, and related molecules, involved in carbohydrate biosynthesis, were shown to be ligands for KIAA0001. Zhu *et al.* (2001)<sup>16</sup> predicted and confirmed not only the ligand but also the pharmacology of a novel GPCR for histamine via phylogeny. Thus phylogeny reconstruction is a significant compute-intensive task within research and developments units of pharmaceutical and biotech companies.

#### 4. COMPUTATIONAL PHYLOGENETICS

Phylogenies have been reconstructed “by hand” for over a century by taxonomists, using morphological characters and basic models of the evolution of these characteristics. With the advent of molecular data, however, it has become necessary to develop algorithms to reconstruct phylogenies from the very large amount of data made available through DNA sequencing, amino-acid and protein characterization, gene expression data, and whole-genome descriptions.

Until recently, most phylogenetic algorithms focused on the development of methods of phylogeny reconstruction from DNA sequences (which can be regarded as strings on a 4-character alphabet), using a model of evolution based mostly on nucleotide substitution. Because amino acids, the building blocks of proteins, are encoded by triplet substrings of four nucleotides known as codons, the same methods were naturally extended to sequences of amino acids (which can be regarded as strings on an alphabet of 22 characters—in spite of the 64 possible codons, only 22 amino acids are encoded, with many codons encoding a single amino acid). Proteins, the natural next level of characterization, are difficult to encode as character states. Recently, another type of data has become available through the characterization of entire genomes: gene content and gene order data.

For some organisms, such as human, mouse, fruit fly, several plants and microorganisms, as well as for a large collection of organelles (mitochondria, the “energy factories” of plants and animals and chloroplasts, the “photosynthesis factories” of plants), we have a fairly complete description of the order of genes of entire genome. Plausible mechanisms of evolution at the “genome level” include gene losses, duplications and rearrangements of the genome. Because evolution at the “genome level” is much slower than mutation driven by mutations in nucleotides (the “gene level”), “genome level” characteristics may help us resolve ancient evolutionary relationships. This new source of data has therefore been embraced by biologists interested in the evolution of major divisions of plants, animals and microorganisms<sup>17-19</sup> despite the fact that it requires models of change that are considerably more complex than those for nucleotide and amino acid sequence data. Thus there has been considerable interest in the interest in the phylogeny community in development of algorithms for reconstructing phylogenies based on gene content and order.

To date, almost every model of change proposed to underlie phylogenetic reconstruction has given rise to NP-hard optimization problems. Three major types of work have arisen: more or less *ad hoc* heuristics (a natural consequence of the

NP-hardness of the problems) that run quickly, but offer no quality guarantees and may not even have a well defined optimization criterion, such as the popular *neighbor-joining* heuristic<sup>20</sup>; optimization problems based on a *parsimony* criterion, which seeks the phylogeny with the least total amount of change needed to explain modern data (a modern version of Occam's razor); and optimization problems based on a *maximum likelihood* criterion, which seeks the phylogeny that is the most likely (under some suitable statistical model) to have given rise to the modern data. *Ad hoc* heuristics are fast and often rival the optimization methods in terms of accuracy; parsimony-based methods may take exponential time, but, at least for DNA data, can often be run to completion on datasets of moderate size; while methods based on maximum-likelihood are very slow (the point estimation problem alone appears intractable) and so restricted to very small instances, but appear capable of outperforming the others in terms of the quality of solutions. In the case of gene-order data, however, only parsimony criteria have been proposed so far: we do not yet have detailed enough models (or ways to estimate their parameters) for using a maximum-likelihood approach.

## 5. HIGH-PERFORMANCE APPROACHES IN DISCRETE ALGORITHMS

The term “algorithm engineering” was first used with specificity in 1997, with the organization of the first *Workshop on Algorithm Engineering (WAE 97)*. Since then, this workshop has taken place every summer in Europe. A parallel workshop was started in the US in 1999, the *Workshop on Algorithm Engineering and Experiments (ALENEX99)*, which has taken place every winter, colocated with the *ACM/SIAM Symposium on Discrete Algorithms (SODA)*. Algorithm engineering refers to the process required to transform a pencil-and-paper algorithm into a robust, efficient, well tested, and easily usable implementation. Thus it encompasses a number of topics, from modeling cache behavior to the principles of good software engineering; its main focus, however, is experimentation. In that sense, it may be viewed as a recent outgrowth of *Experimental Algorithmics*, which is specifically devoted to the development of methods, tools, and practices for assessing and refining algorithms through experimentation. The online *ACM Journal of Experimental Algorithmics (JEA)*, at URL [www.jea.acm.org](http://www.jea.acm.org), is devoted to this area.

High-performance algorithm engineering focuses on one of the many facets of algorithm engineering. The high-performance aspect does not immediately imply parallelism; in fact, in any highly parallel task, most of the impact of high-performance algorithm engineering tends to come from refining the serial part of the code. For instance, in the example we will use in the next section, the million-fold speed-up was achieved through a combination of a 512-fold speedup due to parallelism (one that will scale to any number of processors) and a 2,000-fold speedup in the serial execution of the code. (For more details on high-performance algorithm engineering as it applies to computational biology, see.<sup>21</sup> )

All of the tools and techniques developed over the last five years for algorithm engineering are applicable to high-performance algorithm engineering. However, many of these tools need further refinement. For example, cache-aware programming is a key to performance (particularly with high-performance machines, which have deep memory hierarchies), yet it is not yet well understood, in part through lack of suitable tools (few processor chips have built-in hardware to gather statistics on the behavior of caching, while simulators leave much to be desired) and in part because of complex machine-dependent issues (recent efforts at cache-independent algorithm design<sup>22,23</sup> may offer some new solutions). As another example, profiling a running program offers serious challenges in a serial environment (any profiling tool affects the behavior of what is being observed), but these challenges pale in comparison with those arising in a parallel or distributed environment (for instance, measuring communication bottlenecks may require hardware assistance from the network switches or at least reprogramming them, which is sure to affect their behavior).

## 6. AN ILLUSTRATION: A HIGH-PERFORMANCE SOFTWARE SUITE FOR RECONSTRUCTING PHYLOGENIES FROM GENE-ORDER DATA

### 6.1. Gene-Order Data

Phylogenetic reconstruction from gene-order data is a fairly new endeavor, because it is only recently that a large number of genomes have been fully mapped at the gene level. Currently, most such data sets are derived from mitochondrial and chloroplast genomes. Chloroplasts and mitochondria are organelles with (usually) a single chromosome that live within plant and animal cells. They tend to have small genomes (typically 37 genes for the mitochondria of multicellular animals and approximately 120 genes for the chloroplasts of higher plants). The gene content of these genomes is fairly constant across a wide phylogenetic range; for example, humans, mouse, and fruit fly mitochondria have the same gene content, but differ in their gene order (a small rearrangement between human and mouse and a much more complex rearrangement between these and fruit flies). Note also that genes are directional—that is, they are transcribed in a single direction—so that a gene may change polarity in addition to changing its place in serial order. Because of this, biologists represent gene order along a chromosome as an ordering of signed integers, with each integer representing a single gene. The evolutionary processes that operate on gene order

(as opposed to gene content) include inversions, transpositions, and inverted transpositions. Each of these processes requires two or three breakages in the chromosome and subsequent repair with a placement error; for instance, if the chromosome breaks in two places, the fragment between the breaks may be reattached with the ends switched, creating an inversion.

## 6.2. Approaches to Phylogeny Reconstruction from Gene Order Data

A natural optimization problem for phylogeny reconstruction from gene order data is to reconstruct the most parsimonious tree—the phylogenetic tree with the minimum number of permitted evolutionary events (inversions, transpositions, and inverted transpositions). For any choice of permitted events, such a problem is very computational intensive (known or conjectured to be NP-hard); worse, to date, no good algorithms (efficient or not) exist for solving such problems. Another approach is first to estimate leaf-to-leaf distances (based upon some metric) between all genomes, and then to use a standard distance-based heuristic such as *neighbor-joining*<sup>20</sup> to construct the tree. Such approaches are quite fast and may prove valuable in reconstructing the underlying tree, but cannot recover the ancestral gene orders. A third approach is to encode the gene-order data as sequences of characters and use standard parsimony methods to reconstruct a tree from these sequences.<sup>24,25</sup>

Blanchette *et al.*<sup>26</sup> developed a direct approach, which they called *breakpoint phylogeny*, for the special case in which the genomes all have the same set of genes and each gene appears once. This special case is of interest to biologists, who hypothesize that inversions (which can affect gene order but not gene content) are the primary evolutionary mechanism for a range of genomes or subgenomes (chloroplast and mitochondrial genomes, the non-recombining portions of sex chromosome, etc.) Simulation studies we conducted suggested that this approach works well for certain datasets (i.e., it obtains trees that are close to the model tree), but that the implementation developed by Sankoff and Blanchette, the *BPAnalysis* software,<sup>1</sup> is too slow to be used on anything other than small datasets with a few genes.<sup>24,25</sup>

## 6.3. Breakpoint Analysis

When each genome contains the same set of genes and each gene appears exactly once, a genome can be described by an ordering (circular or linear) of these genes, with each gene being given an orientation that is either positive ( $g_i$ ) or negative ( $-g_i$ ). 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  $(-g_j, -g_i)$  appears consecutively in that order in  $G'$ . The breakpoint distance between two genomes is the number of breakpoints between that pair of genomes. The breakpoint score of a tree in which each node is labelled by a signed ordering of genes is then the sum of the breakpoint distances along the edges of the tree.

Given three genomes, we define their *median* to be a fourth genome that minimizes the sum of the breakpoint distances between it and the other three. The *Median Problem for Breakpoints* (MPB) is to construct such a median and is NP-hard.<sup>27</sup> Sankoff and Blanchette developed a reduction from MPB to the Traveling Salesman Problem (TSP), perhaps the most studied of all optimization problems.<sup>28</sup> Their reduction produces an undirected instance of the TSP from the directed instance of MPB by the standard technique of representing each gene by a pair of cities connected by an edge that must be included in any solution.

*BPAnalysis* (see Figure 2) is the method developed by Blanchette and Sankoff to solve the breakpoint phylogeny. Within

```

For all tree topologies do
  Initially label all internal nodes with gene orders
  Repeat
    For each internal node  $v$ , with neighbors  $A$ ,  $B$ , and  $C$ , do
      Solve the MPB on  $A, B, C$  to yield label  $m$ 
      If relabeling  $v$  with  $m$  improves the score of  $T$ , then do it
    until no internal node can be relabeled
  Return the best tree found

```

**Figure 2.** *BPAnalysis*

a framework that enumerates all trees, it uses an iterative heuristic to label the internal nodes with signed gene orders. This procedure is computationally very intensive. The outer loop enumerates all  $(2n - 5)!!$  leaf-labelled trees on  $n$  leaves, an

exponentially large value.\* The inner loop runs an unknown number of iterations (until convergence), with each iteration solving an instance of the TSP (with a number of cities equal to twice the number of genes) at each internal node. The computational complexity of the entire algorithm is thus exponential in *each* of the number of genomes and the number of genes, with significant coefficients. The procedure nevertheless remains a heuristic: even though all trees are examined and each MPB problem solved exactly, the tree-labeling phase does not ensure optimality unless the tree has only three leaves.

#### 6.4. Re-Engineering BPAanalysis for Speed

**Profiling** Algorithmic engineering suggests a refinement cycle in which the behavior of an implementation is studied so that problem areas, including excessive resource consumption and poor results, can be identified. We used extensive profiling and testing throughout our development cycle, which allowed us to identify and eliminate a number of such problems. For instance, converting the MPB into a TSP instance dominates the running time whenever the TSP instances are not too hard to solve. Thus we lavished much attention on that routine, down to the level of hand-unrolling loops to avoid modulo computations and allowing reuse of intermediate expressions; we cut the running time of that routine down by a factor of at least six—and thereby nearly tripled the speed of the overall code. We lavished equal attention on distance computations and on the computation of the lower bound, with similar results. Constant profiling is the key to such an approach, because the identity of the principal “culprits” in time consumption changes after each improvement, so that attention must shift to different parts of the code during the process—including revisiting already improved code for further improvements. These steps provided a speed-up by one order of magnitude on the *Campanulaceae* dataset.

**Cache Awareness** The original *BPAanalysis* is written in C++ and uses a space-intensive full distance matrix, as well as many other data structures. It has a significant memory footprint (over 60MB when running on the *Campanulaceae* dataset) and poor locality (a working set size of about 12MB). Our implementation has a tiny memory footprint (1.8MB on the *Campanulaceae* dataset) and good locality (all of our storage is in arrays preallocated in the main routine and retained and reused throughout the computation), which enables it to run almost completely in cache (the working set size is less than 600KB). Cache locality can be improved by returning to a FORTRAN-style of programming, in which storage is static, in which records (structures/classes) are avoided in favor of separate arrays, in which simple iterative loops that traverse an array linearly are preferred over pointer dereferencing, in which code is replicated to process each array separately, etc. While we cannot measure exactly how much we gain from this approach, studies of cache-aware algorithms<sup>29–34</sup> indicate that the gain is likely to be substantial—factors of anywhere from 2 to 40 have been reported. New memory hierarchies show differences in speed between cache and main memory that exceed two orders of magnitude.

**Low-Level Algorithmic Changes** Unless the original implementation is poor (which was not the case with *BPAanalysis*), profiling and cache-aware programming will rarely provide more than two orders of magnitude in speed-up. Further gains can often be obtained by low-level improvement in the algorithmic details. In our phylogenetic software, we made two such improvements. The basic algorithm scores every single tree, which is clearly very wasteful; we used a simple lower bound, computable in linear time, to enable us to eliminate a tree without scoring it. On the *Campanulaceae* dataset, this bounding eliminates over 95% of the trees without scoring them, resulting in a five-fold speed-up. The TSP solver we wrote is at heart the same basic include/exclude search as in *BPAanalysis*, but we took advantage of the nature of the instances created by the reduction to make the solver much more efficient, resulting in a speed-up by a factor of 5–10. These improvements all spring from a careful examination of exactly what information is readily available or easily computable at each stage and from a deliberate effort to make use of all such information.

#### 6.5. A High-Performance Implementation

Our resulting implementation, *GRAPPA*,<sup>†</sup> incorporates all of the refinements mentioned above, in addition to others specifically implemented to enable the code to run efficiently in parallel (see<sup>2</sup> for details). Because the basic algorithm enumerates and independently scores every tree, it presents obvious parallelism: a subset of the trees can be assigned to each processor. In order to do this efficiently, we need to impose a linear ordering on the set of all possible trees and devise a generator that can start at an arbitrary point along this ordering. Because the number of trees is so large, an arbitrary tree index would require unbounded-precision integers, considerably slowing down tree generation. Our solution was to design a tree generator that starts with tree index  $k$  and generates trees with indices  $\{k + cn \mid n \in \mathbb{N}\}$ , where  $k$  and  $c$  are regular integers, all without using

\*The double factorial is a factorial with a step of 2, so we have  $(2n - 5)!! = (2n - 5) \cdot (2n - 7) \cdot \dots \cdot 3$

<sup>†</sup>Genome Rearrangement Analysis through Parsimony and other Phylogenetic Algorithms

unbounded-precision arithmetic. Such a generator allows us to sample tree space (a very useful feature in research) and, more importantly, allows us to use a cluster of  $c$  processors, where processor  $i$ ,  $0 \leq i \leq c - 1$ , generates and scores trees with indices  $\{i + cn \mid n \in N\}$ .

The University of New Mexico's Albuquerque High Performance Computing Center operates the Alliance 512-processor supercluster, called *LosLobos* (shown in Figure 3). This platform is a cluster of 256 IBM Netfinity 4500R nodes, each with



**Figure 3.** University of New Mexico's LosLobos SuperCluster

dual 733 MHz Intel Pentium III processors and 1 GB RAM, interconnected by Myrinet 2000 switches. *LosLobos* runs the Linux operating system, and our experiments use the GNU C compiler version egcs-2.91.66 optimized for Pentium Pro (-O3 -mpentiumpro), and MPICH v. 1.1.13 libraries for message passing over the Myrinet with Myricom GM v. 1.3.0 drivers.

We ran *GRAPPA* on *LosLobos* and obtained a 512-fold speed-up (linear speedup with respect to the number of processors): a complete breakpoint analysis (with inversion distances) for the 13 genomes in the Campanulaceae data set ran in less than 1.5 hours. When combined with the 2000-fold speedup obtained through algorithm engineering, our run on the Campanulaceae dataset demonstrated a *million-fold* speed-up over the original implementation.<sup>3</sup>

In addition, we made sure that gains held across a wide variety of platforms and compilers: we tested our code under Linux, FreeBSD, Solaris, and Windows, using compilers from GNU, the Portland group, Intel (beta release), Microsoft, and Sun, and running the resulting code on Pentium- and Sparc-based machines. While the gcc compiler produced marginally faster code than the others, the performance we measured was completely consistent from one platform to the other.

## 7. IMPACT IN COMPUTATIONAL BIOLOGY

Computational biology includes numerous complex optimization problems, such as multiple sequence alignment, phylogeny reconstruction, characterization of gene expression, structure prediction, etc. In addition, the necessary use of very large databases gives rise to serious algorithmic engineering problems with respect to query algorithm design. Although several programs in use (such as BLAST, see [www.ncbi.nlm.nih.gov/BLAST/](http://www.ncbi.nlm.nih.gov/BLAST/)) have already been engineered for performance, most such efforts have been more or less *ad hoc*. The emergent discipline of algorithm engineering<sup>35</sup> is providing tools and standards of practice that can be applied to almost any existing algorithm or software package to speed up its execution, often by significant factors. When these tools and practices are complemented by high-performance software implementations designed for parallel platforms, enormous gains (our example shows six orders of magnitude) may result. Although we illustrated this approach and its potential results with a specific program in phylogeny reconstruction based on gene order data, we are now in the process of applying the same to a collection of fundamental methods (such as branch-and-bound parsimony or maximum-likelihood estimation) as well as new algorithms.

Of course, even large speed-ups have only limited benefits in theoretical terms when applied to NP-hard optimization problems: even our million-fold speed-up with *GRAPPA* only enables us to move from about 10 taxa to 14 taxa. Yet the exploration entailed by algorithm engineering may uncover salient characteristics and dependencies of the algorithm which have

been overlooked in a less careful analysis. In our case, while we were implementing the rather complex algorithm of Berman and Hannenhalli for computing the inversion distance between two signed permutations, which had not been implemented before, we realized that the algorithm could be simplified as well as accelerated, leading us to derive the first true linear-time algorithm for computing these distances.<sup>36</sup> Had we not taken an algorithm engineering approach, we might have been tempted to implement this algorithm in the context of the original program, which was already too slow to be useful in the context of breakpoint distances. Thus faster experimental tools, even if they prove incapable of scaling to “industrial-sized” problems, nevertheless provide crucial opportunities for exploring and understanding a problem and its solutions.

Thus we see two major impacts of algorithm engineering in computational biology. First, the much faster implementations, when mature enough, can alter the practice of research in biology and medicine. For instance pharmaceutical companies spend large budgets on computing equipment and research personnel to reconstruct phylogenies as a vital tool in drug discovery, yet may still have to wait a year or more for the results of certain computations; reducing the running time of such analyses from a couple of years down to a day would make a significant difference in the cost and pace of drug discovery and development. Secondly, biologists in academic laboratories around the world use undocumented heuristics for phylogenetic analysis. Even these heuristics are slow for the purpose for which they were designed. Software that produces solutions with known qualities (such as approximation guarantees) and runs several orders of magnitude faster, even when it remains impractical for real-world problems, would nevertheless enable these researchers to test simpler scenarios, compare models, develop intuition on small instances, and perhaps even form serious conjectures about biological mechanisms.

### ACKNOWLEDGMENTS

This work is supported in part by NSF grants CAREER 00-93039 (Bader) and ITR 00-81404 (Moret and Bader). We thank Tandy Warnow (Dept. of Computer Sciences, U. of Texas at Austin) for getting us started in the area and for her collaboration on many projects in computational phylogenetics and Robert K. Jansen (Section of Integrative Biology, U. of Texas at Austin) for providing the Campanulaceae dataset. We also thank David Klepacki (IBM ACTC group) for discussions on optimizing algorithms for *LosLobos*, our IBM Netfinity Supercluster.

### REFERENCES

1. D. Sankoff and M. Blanchette, “Multiple genome rearrangement and breakpoint phylogeny,” *Journal of Computational Biology* **5**, pp. 555–570, 1998.
2. B. M. Moret, S. Wyman, D. Bader, T. Warnow, and M. Yan, “A new implementation and detailed study of breakpoint analysis,” in *Proceedings of the 6th Pacific Symposium on Biocomputing (PSB2001)*, pp. 583–594, (Big Island, HI), January 2001.
3. D. Bader and B. M. Moret, “GRAPPA runs in record time,” *HPCwire* **9**, November 23 2000.
4. GenProbe, “Detection of *Mycobacterium kansasii* - using a new nucleic acid hybridisation probe with a sequence corresponding to the 23S mRNA variable region of *M. kansasii*.” Patent filing US5681698-A, 1991. (priority date).
5. GenProbe, “Oligo-nucleotide(s) corresp. to *Mycobacterium tuberculosis* sequences - used as probes in hybridisation assays for *M.tuberculosis* Complex.” Patent filing WO9322330-A, 1992. (priority date).
6. B. Engelen, K. Meinken, F. von Wintzingerode, H. Heuer, H.-P. Malkomes, and H. Backhaus, “Monitoring impact of a pesticide treatment on bacterial soil communities by metabolic and genetic fingerprinting in addition to conventional testing procedures,” *Appl. Environ. Microbiol.* **64**, pp. 2814–2821, 1998.
7. P. Halbur, M. Lum, X. Meng, I. Morozov, and P. Paul, “New porcine reproductive and respiratory syndrome virus DNA - and proteins encoded by open reading frames of an Iowa strain of the virus; are used in vaccines against PRRSV in pigs.” Patent filing WO9606619-A1, 1994. (priority date).
8. GenProbe, “New oligonucleotides corresponding to HIV-1 sequences - used for selective amplification and as hybridisation probes for detection of HIV-1.” Patent filing EP-617132-A, 1993. (priority date).
9. R. Overbeek, N. Larsen, G. Pusch, M. D’Souza, E. Selkov Jr, N. Kyrpides, M. Fonstein, N. Maltsev, and E. Selkov, “WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction,” *Nucleic Acids Res.* **28**, pp. 123–125, Jan. 2000.
10. E. Grossbard and D. Atkinson, eds., *The Herbicide Glyphosate*, Butterworths, Boston, MA, 1985.
11. F. Roberts, C. Roberts, J. Johnson, D. Kyle, T. Krell, J. Coggins, G. Coombs, W. Milhous, S. Tzipori, D. Ferguson, D. Chakrabarti, and R. McLeod, “Evidence for the shikimate pathway in apicomplexan parasites,” *Nature*, pp. 801–805, 1998.
12. J. Brown and P. Warren, “Antibiotic discovery: Is it in the genes?,” *Drug Discovery Today* **3**, pp. 564–566, 1998.



13. E. Wilding, J. Brown, A. Bryant, A. Chalker, D. Holmes, K. Ingraham, C. So, S. Iordanescu, M. Rosenberg, and M. Gwynn, "Identification, essentiality and evolution of the mevalonate pathway for isopentenyl diphosphate biosynthesis in Gram-positive cocci," *J. Bacteriology* **182**, pp. 4319–4327, 2000.
14. P. Szekeres, A. Muir, L. Spinage, J. Miller, S. Butler, A. Smith, G. Rennie, P. Murdock, L. Fitzgerald, H. Wu, L. McMillan, S. Guerrero, L. Vawter, N. Elshourbagy, J. Mooney, D. Bergsma, S. Wilson, and J. Chambers, "Neuromedin U is a potent agonist at the orphan G protein-coupled receptor FM3," *J Biol Chem.* **275**(27), pp. 20247–20250, 2000.
15. J. Chambers, L. Macdonald, H. Sarau, R. Ames, K. Freeman, J. Foley, Y. Zhu, M. McLaughlin, P. Murdock, L. McMillan, J. Trill, A. Swift, N. Aiyar, P. Taylor, L. Vawter, S. Naheed, P. Szekeres, G. Hervieu, C. Scott, J. Watson, A. Murphy, E. Duzic, C. Klein, D. Bergsma, S. Wilson, and G. Livi, "A G protein-coupled receptor for UDP-glucose," *J Biol Chem* **275**(15), pp. 10767–10771, 2000.
16. Y. Zhu, D. Michalovich, H. Wu, K. Tan, G. Dytko, I. Mannan, R. Boyce, J. Alston, L. Tierney, X. Li, N. Herrity, L. Vawter, H. Sarau, R. Ames, C. Davenport, J. Hieble, S. Wilson, D. Bergsma, and L. Fitzgerald, "Cloning, expression, and pharmacological characterization of a novel human histamine receptor," *Mol Pharmacol.* **59**(3), pp. 434–441, 2001.
17. R. Olmstead and J. Palmer, "Chloroplast DNA systematics: a review of methods and data analysis," *Am. J. Botany* **81**, pp. 1205–1224, 1994.
18. J. Palmer, "Chloroplast and mitochondrial genome evolution in land plants," in *Cell Organelles*, R. Herrmann, ed., pp. 99–133, Springer Verlag, 1992.
19. L. Raubeson and R. Jansen, "Chloroplast DNA evidence on the ancient evolutionary split in vascular land plants," *Science* **255**, pp. 1697–1699, 1992.
20. N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstruction of phylogenetic trees," *Molec. Biol. Evol.* **4**, pp. 406–425, 1987.
21. B. Moret, D. Bader, and T. Warnow, "High-performance algorithm engineering for computational phylogenetics," in *Proceedings of the 2001 International Conference on Computational Science*, V. Alexandrov, J. Dongarra, and C. Tan, eds., Springer-Verlag, (San Francisco, CA), May 2001. in Lecture Notes in Computer Science.
22. M. Bender, E. Demaine, and M. Farach-Colton, "Cache-oblivious search trees," in *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS-00)*, pp. 399–409, IEEE Press, (Redondo Beach, CA), Nov. 2000.
23. M. Frigo, C. Leiserson, H. Prokop, and S. Ramachandran, "Cache-oblivious algorithms," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS-99)*, pp. 285–297, IEEE Press, (New York, NY), 1999.
24. M. Cosner, R. Jansen, B. Moret, L. Raubeson, L. Wang, T. Warnow, and S. Wyman, "A new fast heuristic for computing the breakpoint phylogeny and a phylogenetic analysis of a group of highly rearranged chloroplast genomes," in *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB00)*, pp. 104–115, (San Diego, CA), 2000.
25. M. Cosner, R. Jansen, B. Moret, L. Raubeson, L. Wang, T. Warnow, and S. Wyman, "An empirical comparison of phylogenetic methods on chloroplast gene order data in Campanulaceae," in *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, D. Sankoff and J. Nadeau, eds., pp. 99–121, Kluwer Academic Pubs., Dordrecht, Netherlands, 2000.
26. M. Blanchette, G. Bourque, and D. Sankoff, "Breakpoint phylogenies," in *Genome Informatics*, S. Miyano and T. Takagi, eds., pp. 25–34, University Academy Press, Tokyo, Japan, 1997.
27. I. Pe'er and R. Shamir, "The median problems for breakpoints are NP-complete," Tech. Rep. Report 71, Electronic Colloquium on Computational Complexity, Nov. 1998.
28. D. Johnson and L. McGeoch, "The traveling salesman problem: a case study," in *Local Search in Combinatorial Optimization*, E. Aarts and J. Lenstra, eds., pp. 215–310, John Wiley, New York, 1997.
29. L. Arge, J. Chase, J. Vitter, and R. Wickremesinghe, "Efficient sorting using registers and caches," in *Proceedings of the 4th Workshop on Algorithm Engineering (WAE 2000)*, Springer-Verlag, (Saarbrücken, Germany), Sept. 2000.
30. N. Eiron, M. Rodeh, and I. Steinwarts, "Matrix Multiplication: A Case Study of Enhanced Data Cache Utilization," *ACM Journal of Experimental Algorithmics* **4**(3), 1999. <http://www.jea.acm.org/1999/EironMatrix/>.
31. R. Ladner, J. Fix, and A. LaMarca, "The cache performance of traversals and random accesses," in *Proc. 10th Ann. ACM/SIAM Symposium on Discrete Algorithms (SODA-99)*, pp. 613–622, (Baltimore, MD), 1999.
32. A. LaMarca and R. Ladner, "The Influence of Caches on the Performance of Heaps," *ACM Journal of Experimental Algorithmics* **1**(4), 1996. <http://www.jea.acm.org/1996/LaMarcaInfluence/>.
33. A. LaMarca and R. Ladner, "The Influence of Caches on the Performance of Heaps," in *Proceedings of the Eighth ACM/SIAM Symposium on Discrete Algorithms*, pp. 370–379, (New Orleans, LA), 1997.

34. L. Xiao, X. Zhang, and S. Kubricht, "Improving memory performance of sorting algorithms," *ACM Journal of Experimental Algorithmics* **5**(3), 2000. <http://www.jea.acm.org/2000/XiaoMemory/>.
35. B. Moret, "Towards a discipline of experimental algorithmics," in *DIMACS Monographs in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, 2001. To appear. Available at [www.cs.unm.edu/~moret/dimacs.ps](http://www.cs.unm.edu/~moret/dimacs.ps).
36. D. Bader, B. M. Moret, and M. Yan, "A fast linear-time algorithm for inversion distance with an experimental comparison," in *Proceedings of the Seventh International Workshop on Algorithms and Data Structures (WADS 2001)*, Springer-Verlag, (Providence, RI), Aug. 2001.