Cladistics 17, S71–S82 (2001)
doi:10.1006/clad.2000.0160, available online at http://www.idealibrary.com on IDEAL®

Efficiency of Parallel Direct Optimization

Daniel A. Janies[1] and Ward C. Wheeler

Division of Invertebrate Zoology, American Museum of Natural History, Central Park West at 79th Street, New York, New York 10024

Accepted December 12, 2000

Tremendous progress has been made at the level of sequential computation in phylogenetics. However, little attention has been paid to parallel computation. Parallel computing is particularly suited to phylogenetics because of the many ways large computational problems can be broken into parts that can be analyzed concurrently. In this paper, we investigate the scaling factors and efficiency of random addition and tree refinement strategies using the direct optimization software, POY, on a small (10 slave processors) and a large (256 slave processors) cluster of networked PCs running LINUX. These algorithms were tested on several data sets composed of DNA and morphology ranging from 40 to 500 taxa. Various algorithms in POY show fundamentally different properties within and between clusters. All algorithms are efficient on the small cluster for the 40-taxon data set. On the large cluster, multibuilding exhibits excellent parallel efficiency, whereas parallel building is inefficient. These results are independent of data set size. Branch swapping in parallel shows excellent speed-up for 16 slave processors on the large cluster. However, there is no appreciable speed-up for branch swapping with the further addition of slave processors (>16). This result is independent of data set size. Ratcheting in parallel is efficient with the addition of up to 32 processors in the large cluster. This result is independent of data set size.

INTRODUCTION

The algorithmics of tree searching have been studied extensively, and most work has focused on methods of initial tree construction and rearrangement (branch swapping) (Farris, 1978, 1988, 1995; Swofford, 1989–1999; Felsenstein, 1980–2000; Goloboff, 1993–2000; Wheeler and Gladstein, 1994–2000; Gladstein and Wheeler, 1996–2000). There has been tremendous improvement in tree search algorithms in the past 2 years. These improvements include the use of genetic algorithms [implemented as natural selection on trees (Moilanen, 1999) and tree fusing (Goloboff, 1999)], sectorial searches (Goloboff, 1999), and simulated annealing [implemented as ratcheting (Nixon, 1999) and tree-drifting (Goloboff, 1999)]. In about the same period of time, workers in disciplines as diverse as filmmaking, oil exploration, genomic contig assembly, particle

[1]To whom correspondence should be addressed. E-mail: djanies@amnh.org. Software available anonymously: ftp://ftp.amnh.org/pub/molecular/poy.

0748-3007/01 $35.00

physics, and astrophysics have begun to take advantage of the power of cluster-computing.[2] However, phylogeneticists have just begun to harness parallelism. Here, we report on the parallel efficiency of various algorithms in POY, software for direct optimization of DNA and morphology (Gladstein and Wheeler, 1996–2000; Janies and Wheeler, 2000, 2001). The algorithms studied include two types of initial cladogram building (parallel building or multibuilding) and several strategies for tree refinement (SPR and TBR branch swapping, parallel ratcheting, and multiratcheting). Algorithms were tested on several data sets composed of DNA and morphology ranging from 40 to 500 taxa. Also a small cluster (10 slave processors) was compared to a large cluster (256 slave processors).

Parallelism, Overhead, and Granularity

The current parallel implementation of POY in PVM (Parallel Virtual Machine; Geist *et al.*, 1993) is simple. A master processor is designated and used to spawn jobs to slave processors. As slave processors complete tasks or portions thereof, slaves return results to the master. POY running locally on the master collates the results, decides on courses of action, and sends new jobs to slaves. This mode of parallelism is termed message passing.

With message passing on standard network infrastructure (e.g., 10 or 100 Mbps Ethernet), overall computation of a large problem can be accelerated appreciably by parallelism only if various subproblems can be computed concurrently and independently, with a minimum of parallel overhead. Overhead consists of bookkeeping by the master processor to manage the message passing and the actual time of interprocessor communication, which is very slow relative to the amount of computation that can be accomplished by each slave processor.

The granularity of the algorithm refers to the size of the subproblem spawned to each process. We investigated the parallel efficiency of algorithms of various granularity within each of the major search steps of POY.

[2]Many researchers are constructing supercomputers from commodity hardware, standard networks, and open-source operating systems. These supercomputers are often termed Beowulf clusters. For a historical overview see Sterling and Savarese (1999). For a technical background see Sterling *et al.*, (1999).

Building

Initial building of each tree in a series of replicates using the cluster as a collective, termed parallel building, requires that each slave processor receives messages from the master on which taxon to add and which subset of places to put that taxon on a subtree (Fig. 1, top) (commands `-parallel -random` *n*; let *n* equal the number of searches, random addition sequences through refinement as specified). Parallel

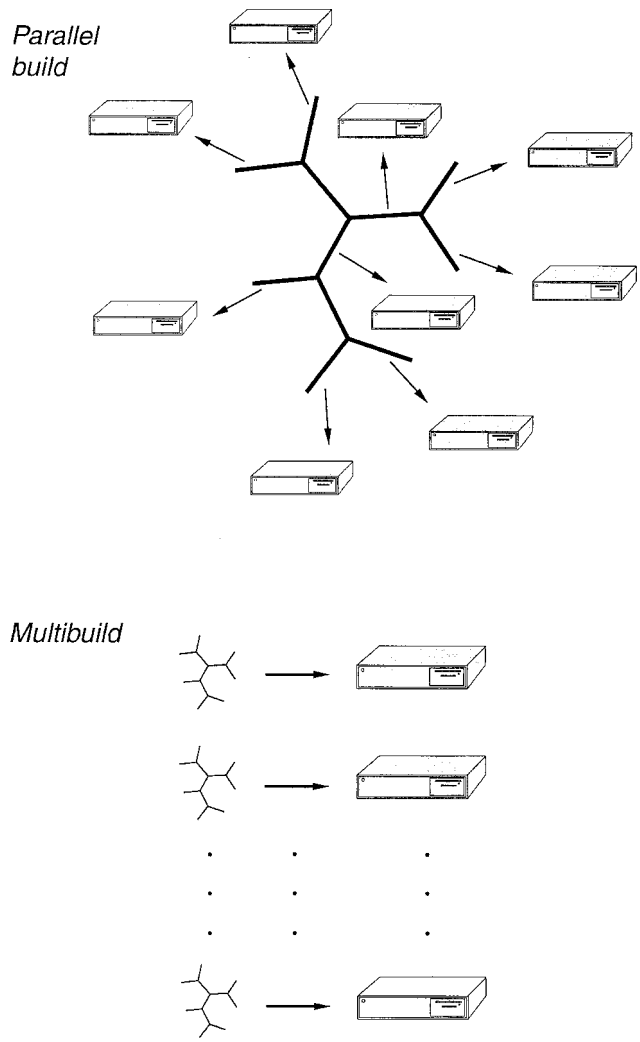

FIG. 1. Contrast of strategy between parallel building (using the cluster as a collective) and multibuilding (using a one-processor-per-replicate strategy). "Multi" strategies have been developed for ratcheting (efficiency of command -multiratchet decribed herein) and for entire searches (e.g., -multirandom). Watch ftp site, listserver, and documentation for additional commands.

building will use a maximum of 2*t*-5 slave processors if each computes a single placement of a taxon (let *t* equal the number of taxa). Over the course of a parallel build, an average of (2*t*-5)/2 slave processors will be employed.

Parallel building is a fine-grained operation. The cost of computation of adding a single taxon is small and expands linearly with the number of characters (Gladstein, 1997; this work also provides some speed-ups that have not yet been implemented in POY). The cost of computation of adding a single taxon is simply a treelength calculation resulting from comparing the character states of the taxon to be added to the union of character states for ancestor and descendents on the branch to be added (Goloboff, 1996). The treelength for each taxon addition must be communicated from each slave back to the master for each addition point for every taxon in a data set. As a result, the ratio of communication relative to computation is very high. Once a build is completed, the next replicate in the series can be addressed.

As an alternative to using the slaves as a collective on a single build, each complete build replicate can be computed on an individual slave processor (commands `-parallel -random` *n* `-multibuild` *m*; let *m* equal the number of random addition sequences) (Fig. 1). The command `-multibuild` *m* causes "*m*" random addition sequence builds (no swapping) to be performed on slave processors. In practice, *m* is often set to the number of available slave processors. The more slave processors available, the more random addition sequences that can be explored; hence, shorter builds are usually discovered. In typical searches, the shortest build(s) is submitted to branch swapping. If `-random` *n* is also specified, the *m* builds will occur for each search resulting in "$n \times m$" builds and "*n*" swapping rounds.

This strategy, termed multibuilding, is a coarse-grained operation with a low communication/computation ratio. In multibuilding, there is no interprocessor communication while taxa are added to various subtrees under independent consideration on each slave. As a slave completes a build, it sends the best topology (topologies) and treelength found back to the master for collation and comparison to results of other builds on other slaves before tree refinement operations begin.

The cost of computation of each initial build is combinatorially expansive for an exact search. The number of possible rooted topologies increases as a power series (where *i* is the starting point and *t* is the number of taxa) let y equal the number of topologies $\prod_{i=2}^{t} (2i-3)$ (Cavalli-Sforza and Edwards, 1967). Multiple sequence alignment along a tree is an NP-complete problem (Wang and Jiang, 1994) and heuristics are required for biologically interesting data sets. For example, if one had 30 taxa representing exemplars of most metazoan phyla, an exact search would require examining over 5×10^{38} trees. The heuristics commonly implemented in phylogenetic algorithms are sensitive to the order in which taxa are accreted into an alignment or tree optimization procedure; thus random replication of the build process is important to adequate searching. In POY, the cost of computation of each build on a single slave processor increases proportionally to $t^{1.96}$ (let *t* equal the number of taxa in the data set let y equal the number of topologies) (Fig. 3a). Multibuilding is a simple means of allowing the user to spread a series of random builds across slave nodes of a cluster. However, as the number of taxa becomes large, the per-slave run times could be prohibitively long.

In subsequent steps of a search, parallel efficiency is limited if initiation of work cannot begin until other subproblems have been computed. For example, once a set of multibuild replicates is complete the master node evaluates the results and sends out the best topology (topologies) for subsequent tree refinement such as branch swapping or ratcheting. In a search such as this, parallel efficiency is often diminished because builds vary in execution time and the results of all replicates are required to proceed to branch swapping and/or ratcheting.

Branch Swapping

In SPR (subtree pruning and redrafting) branch swapping, each taxon and each internal node are removed from the tree and placed back at all other possible positions. In parallelization of SPR, various pairs of subtrees resulting from prunings are passed to individual slaves where the optimality value for each regraft is calculated. Once a set of regraftings is assessed, the best topology (topologies) and treelengths are reported back to the master. SPR involves moderate communication/computation ratios. For SPR in POY on a tree with *t* taxa, there are 2*t*-4 branches to prune off

the tree. A load-balancing problem arises in placing the subtrees back on the tree. The larger the subtree (number of descendants), the smaller the set of regrafting points on the remaining fraction of the tree. The granularity of swapping varies. The number of candidate replacements varies from 3 to 2*t*-3, thus producing a large spread of execution times.

In TBR (tree bisection and reconnection), the number of subtrees pruned is smaller than that in SPR because prunings occur only along internal branches. However, the number of regrafts in TBR is greater than that in SPR because subtrees are derooted and all possible rearrangements are tried. The number of candidate replacements is the product of the number of branches (minus one replacement point that represents the original tree). Thus a load-balancing problem occurs in TBR but it is limited by the number of prunings rather than regrafts as in SPR.

In both modes of branch swapping, the shortest trees are kept and further rounds of rearrangements are tried on the shortest trees until no shorter candidate trees are found. It is not possible to predict how many swapping rounds will be needed as this is tied to heuristics, data set structure, and length of initial cladogram builds. In theory, the cost of computation of SPR should be proportional to the square of the number of taxa and the cost of computation of TBR should be proportional to the cube of the number of taxa. When considering heuristics in POY, the cost of computation for branch swapping is proportional to $t^{1.93}$ for SPR and $t^{2.34}$ for TBR (let t equal the number of taxa) (Figs. 2b and 2c). In practice, both modes of branch swapping are employed by POY users as the default is to use SPR followed by TBR (we examined the efficiency of commands `-parallel, -spr, -tbr`).

Ratcheting

The parsimony ratchet (Nixon, 1999) is a novel method that is extremely efficient at exploring treespace rapidly by searching multiple islands of trees. This method resulted in search times to the shortest known tree for the 500-taxon rbcL data set that were several thousand times faster than those obtained using the more standard methods of random taxon sequence addition, SPR and TBR (Chase, *et al.,* 1993; Nixon, 1999). The method therefore extends the size of data sets that can be analyzed in a reasonable time.

After initial cladogram building and standard branch swapping (if specified) a randomly selected subset of characters is reweighted as specified by the command `-ratchetpercent` *n* (let *n* equal the percentage of characters to be reweighted). A branch swapping search on the current tree begins with the new weights. Two swapping strategies can be used (commands `-ratchetspr` *n*, `-ratchettbr` *n*; let *n* equal the number of iterations). Typically a small number of trees (one or two) is held as specified by the command `-ratchettrees` *n* (let *n* equal the number of trees held). The characters are then reset to the original weights and swapping recommences on the held tree(s). Trees shorter than the original tree might be found. The crux of the method is that reweighted characters might find trees that are potentially not on the same "island" as the current tree. In other words, the original best tree is locally optimal for the original weights but is perhaps not globally optimal in treespace. Areas outside of local optima are investigated to find better local, or potentially globally optimal, solutions.

In POY, ratcheting occurs in parallel using the same code as swapping in parallel described above. However, ratcheting is an iterative method and the cycle described above can be continued for several iterations (commands `-ratchetspr` *n*, `-ratchettbr` *n*). The cost of computation of ratchettbr (Fig. 2d) in POY is proportional to $t^{2.51}$, similar to that of TBR $t^{2.34}$ (let t equal the number of taxa let y equal the number of topologies) (Fig. 2c). However, many more trees are examined with ratchettbr than with TBR (compare the *y* axes of Figs. 2c and 2d).

Just as in the multibuilding strategy described above, some of the simplest, yet most effective, implementations of a parallel cluster are the simultaneous exploration of many replicates, each independently assigned to slave processors. Ratcheting can be parallelized at the level of reweighting individual random subsets of characters and spawning each replicate to a slave via the commands `-multiratchet -ratchettbr` *n* or multiratchet `-ratchetspr` *n* (let *n* equal the number of ratchet replicates to be spawned to slaves). Multiratcheting increases the granularity and reduces communication/computational ratios of ratcheting because larger chunks of work are done by each slave between communication events.

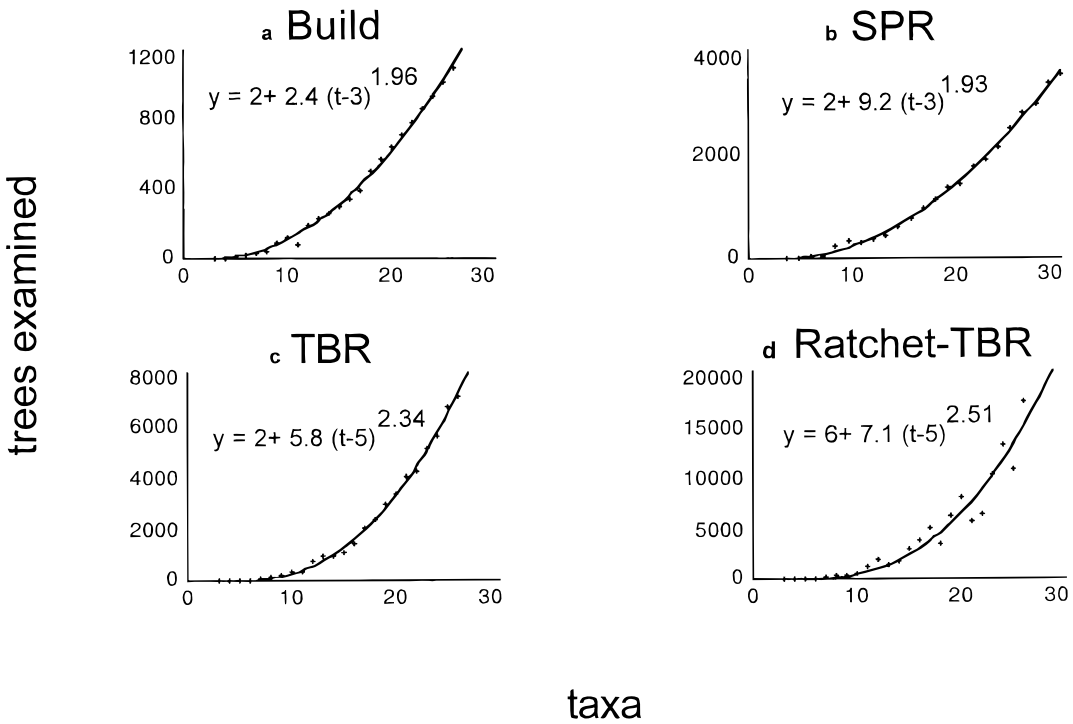

FIG. 2. Cost of computation estimates for POY of (a) tree building, (b) SPR, (c) TBR, and (d) Ratchet-TBR. Cost computation was estimated empirically by fitting curves to the number of trees examined for complete rounds of each operation for data sets of various numbers of taxa but with a consistent number of characters. The effects of data sets of large numbers of taxa and complex structure on parallel efficiency merit further study.

Another very simple, very coarse-grained mode of parallelism can be applied to sensitivity analysis (Wheeler, 1995), under which various character transformation costs and data partitions of a large data set can be assigned to subsets of processors. In fact, this mode of parallelism requires no interprocessor communication if one chooses to use "sneakernet" to walk between a variety of PCs to set up various jobs and collate the results by hand (*e.g.*, D'Haese, 2000).

METHODS

Hardware

Cluster computing is feasible due to inexpensive commodity PC hardware, the prevalence of fast Ethernet (100 Mbps) networks, and the free availability of efficient and stable operating systems, message passing software, and applications. At the American Museum of Natural History, we have constructed parallel computers from ordinary CPUs, motherboards, hard drives, RAM, network cards, Ethernet switches, or hubs. We run LINUX (operating system Kernel 2.2.5–15 smp), PVM (message passing software), POY (direct or fixed state optimization software), or MALIGN [Wheeler and Gladstein, 1994–2000 (multiple alignment software)]. Other phylogenetic software is available for PVM (e.g., fast-DNAml; Ceron *et al.*, 1998). We chose nonproprietary hardware and software to (1) maximize performance per dollar; (2) minimize reliance on ineffective systems and repair staff; and (3) make upgrades and repairs doable by users. Two different PC clusters were used to study the scalability of POY's algorithms. The small cluster was composed of 11 Intel 200 MHz Pentium IIs networked via 10

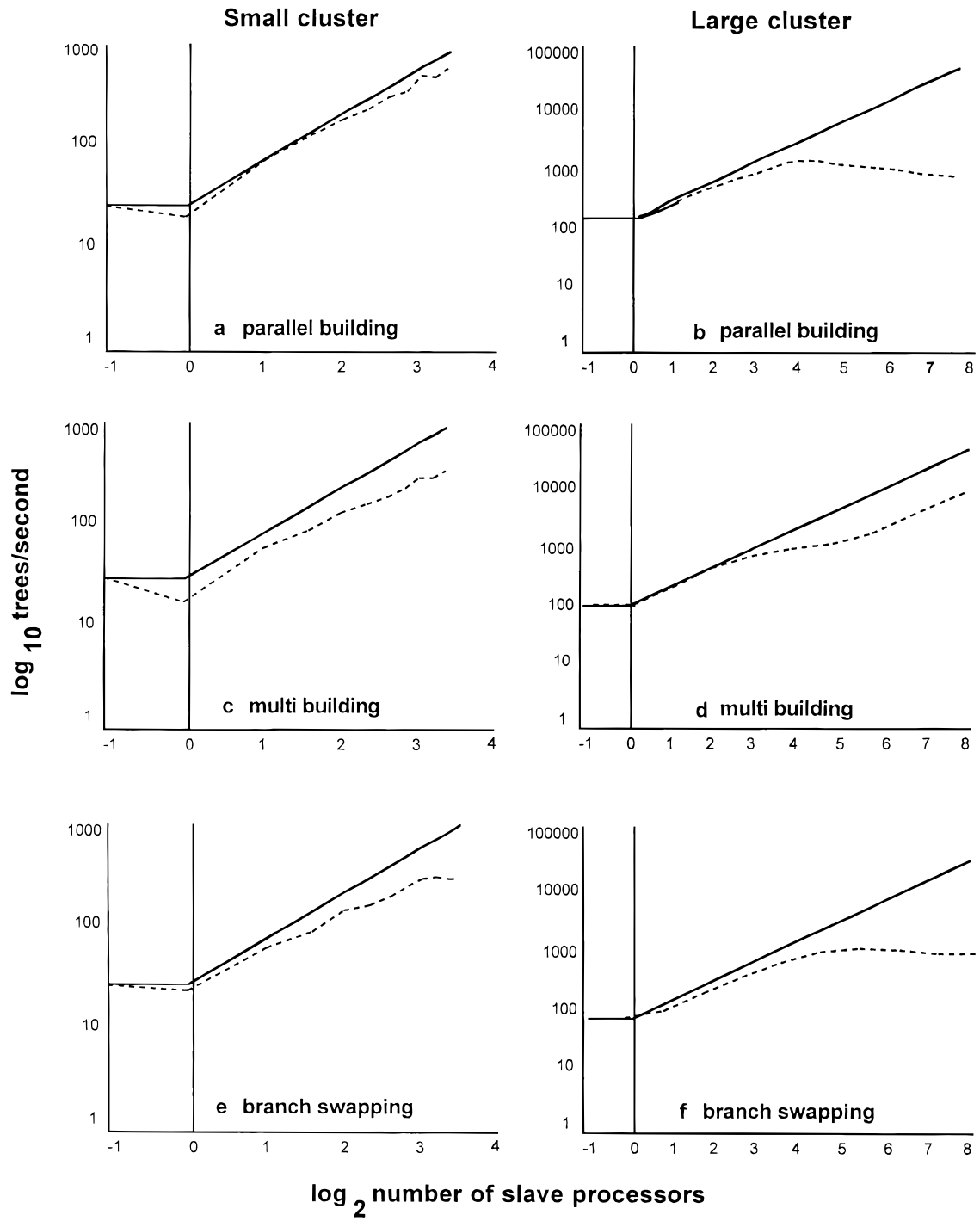

FIG. 3. Graphs of parallel efficiency of parallel building, multibuilding, and branch swapping in POY in two different clusters using the 40-taxon data set. The small cluster was composed of 11 Intel 200-MHz Pentium IIs networked via 10 Mbps Ethernet through a hub. The large cluster is composed of 258 500-MHz Intel Pentium IIIs networked via 100 Mbps Ethernet through a Cisco 5509 switch. The solid lines represent perfect parallel speed-up. The dotted lines represent actual speed-up. The *y* axis is $\log_{10}$ trees examined per second. The *x* axis is

Mbps Ethernet through a hub. The large cluster is composed of 258 500-MHz Intel Pentium IIIs networked via 100 Mbps Ethernet through a Cisco 5509 switch.[3] We have avoided exotic network topologies to provide the simplest possible environment for biological studies, algorithm development, and cluster engineering. This paper is a first examination of these choices and the parallel efficiency of POY. We are using these results to fine-tune the future development of hardware, software, and network topology.

Sample Data Sets

Four data sets were chosen to provide an increasing series in number of taxa. These data sets include the following: 12S rDNA, 16S rDNA, and morphology for 40 phrynosomatid lizard taxa (Reeder, 1995); 18S and 28S rDNA from 130 hexapod taxa (Wheeler *et al.*, in press); 12S rDNA for 264 mammal taxa (Emerson and Allard, in preparation); and rbcL DNA for 500 angiosperm taxa (Chase *et al.*, 1993).

[3]Switches are efficient and intelligent networking devices because they route data directly from the source node to the destination node without creating unnecessary traffic on the network. Hubs are an older technology that inefficiently broadcast to every node on the network to search out the destination node before transmitting the data from node to node.

Runs

All analyses were done while there was no other activity on the cluster. The 40-taxon data set was run on the small cluster but the larger data sets were not. All data sets were run on the large cluster. The amount of computing power applied to each data set was graduated in the small cluster by adding slave processors linearly (1 to 10 slaves) and as a series of eight doublings for the large cluster (1–256 slaves) (command `-maxprocessors` *n*). Major algorithms of POY were tested for parallel efficiency: initial cladogram building, branch swapping, and ratcheting. Each script was written to isolate the behavior of an algorithm rather than execute a standard set of tree search procedures. For example, when building was of interest, swapping and ratcheting were not invoked. Conversely when branch swapping and ratcheting were investigated, an initial topology was specified and cladogram building was disabled. Scripts and data sets are available anonymously (ftp://ftp.amnh.org/pub/people/djanies).

Random replicates of initial cladogram builds were distributed to several processors via a one-processor-per-replicate strategy (commands `-parallel -multibuild` *m*). Alternatively, single-cladogram builds were partitioned across many processors (commands `-parallel`). Branch swapping rounds were partitioned across many processors (commands

---

$\log_2$ number of slave processors to provide a series of doublings. For example:

Small cluster		Large cluster	
Slave processors	$\log_2$ slave processors	Slave processors	$\log_2$ slave processors
0	−1	0	−1
1	0	1	0
2	1	2	1
3	1.58	4	2
4	2	8	3
5	2.32	16	4
6	2.58	32	5
7	2.81	64	6
8	3	128	7
9	3.17	256	8
10	3.32		

The secondary *y* axes represent the point at which parallelism begins (i.e., if $\log_2$ number of slave processors = 0, then there is one slave). Any dip in actual versus expected performance represents the sum of the communications and programming overhead associated with the implentation of message passing parallelism. This secondary *y* axis is omitted in subsequent figures for simplicity.

`-parallel -tbr -spr`). Single-ratchet replicates were partitioned across many processors (commands `-parallel -ratchettbr` *r*; let *r* equal the number of ratchet replicates) or each replicate was assigned a single slave processor (commands `-parallel -multiratchet -ratchettbr` *r*). Slight variability in run time due to heuristics employed in these algorithms was not corrected because the trends in the run-time data are more important than the actual numbers.

RESULTS

Various algorithms in POY show fundamentally different properties within and between clusters. The results for parallel building and branch swapping for the 40-taxon lizard data set on the large cluster contrast with results for the small cluster for parallel building and swapping. Parallel building exhibits excellent speed-up (trees examined per second) for the 40-taxon data set in the small cluster (up to 10 slave processors) with a slight initial overhead with the addition of the first slave processor (Fig. 3a). Parallel building is efficient for the 40-taxon data set in the large cluster to 16 processors with a slight initial overhead but no additional speed-up is gained with additional processors (Fig. 3b).

Multibuilding exhibits parallel efficiency for the small cluster for the 40-taxon data set but with an initial overhead for adding the first slave processor (Fig. 3c). Speed-up under multibuilding is linear for up to 10 slave processors of the small cluster but speed-up refracts beyond 16 slave processors with the large cluster (Fig. 3d).

Parallel building is not efficient for the three large data sets (130 hexapods, 264 mammals, and 500 angiosperms) for the large cluster (Figs. 4a, 4c, and 4e). This result is slightly dependent on data set size with some speed-up for the hexapod data set to 64 processors but almost no speed-up for any of the other large data sets.

Multibuilding exhibits excellent parallel efficiency for the three large data sets on the large cluster (Figs. 4b, 4d, and 4f). Speed-up is very close to linear with the addition of processors regardless of data set or cluster size.

SPR branch swapping and TBR branch swapping show good speed-up for the 40-taxon data set for up to 8 slave processors on the small cluster but the curve flattens with the addition of slave processors (Fig. 3e). Speed-up for branch swapping refracts beyond the addition of 16 slave processors on the large cluster for all data sets (Figs. 3f, 5a, 5b, and 5c).

Ratcheting (Nixon, 1999) is implemented in branch swapping. The parallel efficiency of partitioning each ratchet job across the slave processors of the large cluster is similar to that in branch swapping. Ratcheting (`-parallel -ratchettbr` *n*) shows good speed-up with the addition of up to 16 processors and this is independent of data set size (Figs. 6a, 6c, and 6e). However, unlike branch swapping, there are lags in speed-up upon the initiation of parallel ratcheting in the 130-hexapod data set (Fig. 6a) and the 264-mammal data set (Fig. 6c). This lag does not occur in the 500-angiosperm data set (Fig. 6e).

With the modification of the parallel behavior of ratcheting such that each replicate is assigned a single slave processor (commands `-parallel -ratchettbr` *n* `-multiratchet`) ratcheting is very efficient and fast for the three large data sets on the large cluster (Figs. 6b, 6d, and 6f).

DISCUSSION

Parallel building is efficient in small clusters for small data sets. The efficiency of parallel building in the large cluster is poor irrespective of data set size. However, this strategy may work well if the number of taxa greatly exceeds the number of slave processors in the cluster. Nevertheless, parallel building will certainly involve a large number of communication events and will require fast switches.

The lack of speed-up of multibuilding with the addition of more than 16 processors on the 40-taxon data set in the large cluster is counterintuitive. One explanation is that the refraction at 16 processors in the large cluster is a manifestation of the overhead seen after the addition of only one slave processor for multibuilding on the 40-taxon data set on the small cluster. The small cluster has 10 Mbps Ethernet running via a hub, whereas the large cluster has 100 Mbps switched Ethernet; thus the overhead lag did not manifest itself until there was moderate master–slave communication

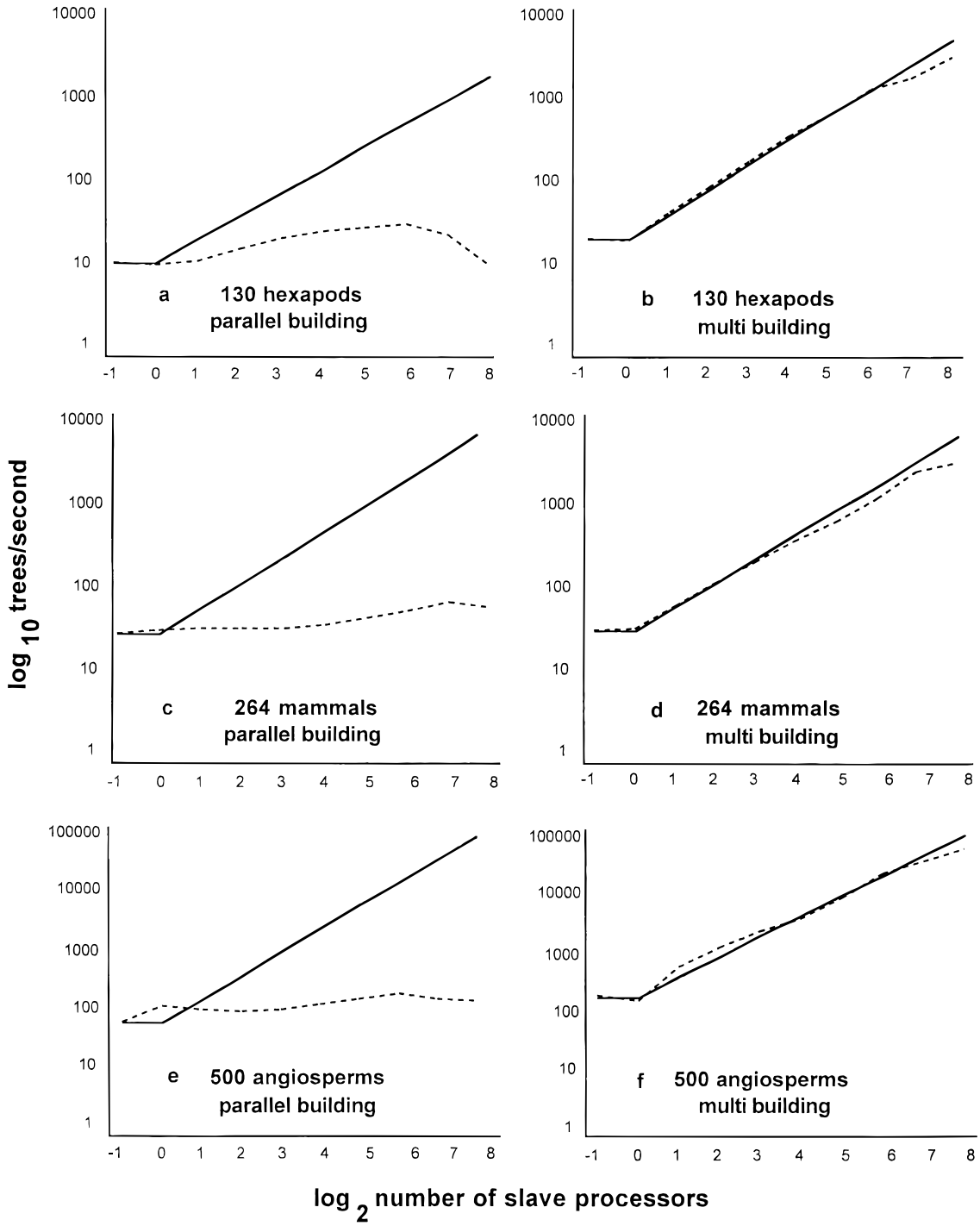

FIG. 4. Graphs of parallel efficiency of parallel building and multibuilding on the large cluster for three data sets: 130 hexapods, 264 mammals, and 500 angiosperms. Parallel building is inefficient and multibuilding is efficient to 256 processors regardless of data set size.

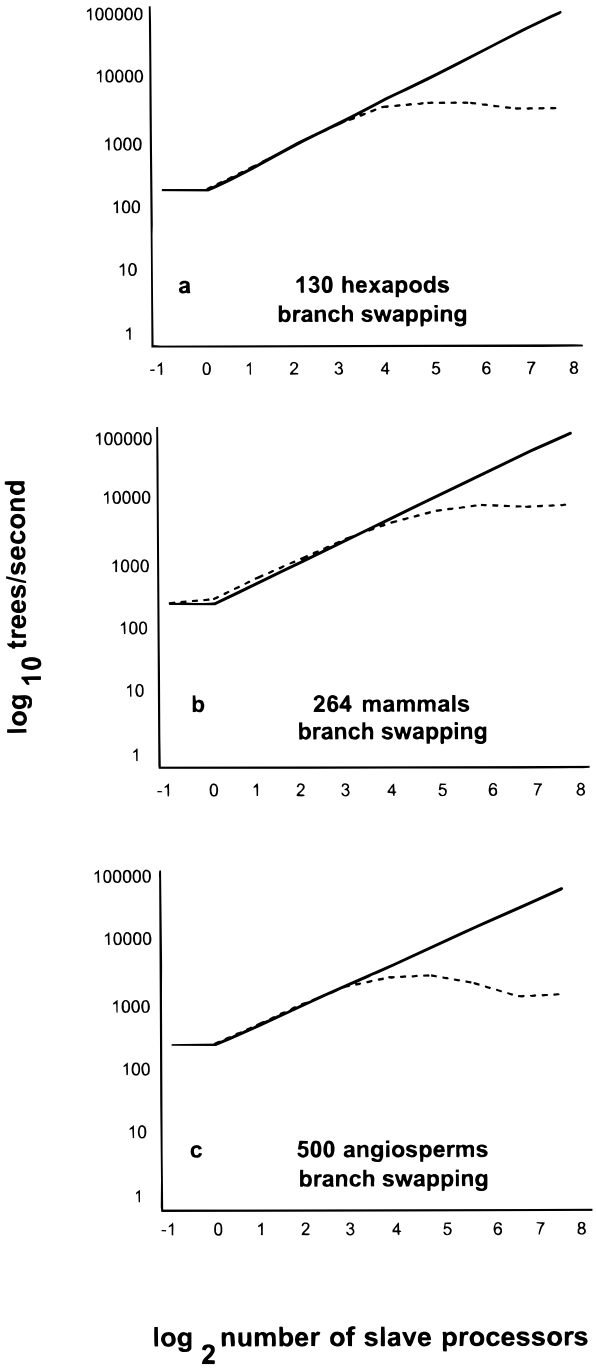

FIG. 5. Graphs of parallel efficiency of branch swapping on the large cluster for three data sets: 130 hexapods, 264 mammals, and 500 angiosperms. Branch swapping is very efficient to 16 processors but not beyond. This result is independent of data set size.

(Fig. 3d). The inefficiency on the large cluster of multibuilding of small (Fig. 3d) versus large data sets (Figs. 4b, 4d, and 4f) is likely due to the increase in the communication/computation ratio with small data sets (i.e., the per-slave run time is short).

These results are fundamental to improving the algorithms for multiuser load balancing and to achieve maximum performance per unit investment. The excellent parallel efficiency of the multibuild and multiratchet algorithms is very encouraging. These results demonstrate the viability of building clusters composed of hundreds of processors without investing in expensive, nonstandard, network hardware. Also, the success of one-replicate-per-slave strategies, such as multibuild or multiratchet, forces a shift in allocation of monetary resources from networking infrastructure to higher clockspeed processors to shorten per-processor run times. We are currently prototyping faster slave processors and faster network topologies and protocols. Work is under way to develop algorithms that are adaptive and have multiple layers of parallelism. Future versions of POY will adjust each step of a search based on knowledge of algorithmic efficiency and monitoring of efficiencies and user load. One way to accomplish this is to spawn jobs hierarchically by designating master processors, submasters, and slaves. For example, we found that branch swapping is efficient to about 16 processors. With this knowledge a large cluster of 256 processors will be divided into 16 teams of 16 slaves under the control of submasters and each team will be assigned swapping replicates. With more efficient and user-transparent algorithms, and continued performance/price increases in CPU products, we predict that these techniques and hardware will become commonplace in evolutionary biology and genomics in a very short period of time.

ACKNOWLEDGMENTS

The National Aeronautics and Space Administration—Ames Research Center (NAG 2-1399 to Janies and Wheeler and NAG 2-1399 to Wheeler and MacLow), the American Museum of Natural History, and the New York City Department of Cultural Affairs provided research funding. Lisa Gugenheim, Tim Mohrmann, Pete Makovicky, Diego Pol, Estelle Perrera, Al Phillips, Julian Faivovich, Rebecca Klasfeld, Mike Benedetto and Mario Reed of the AMNH were instrumental in the procurement and construction of the parallel cluster.

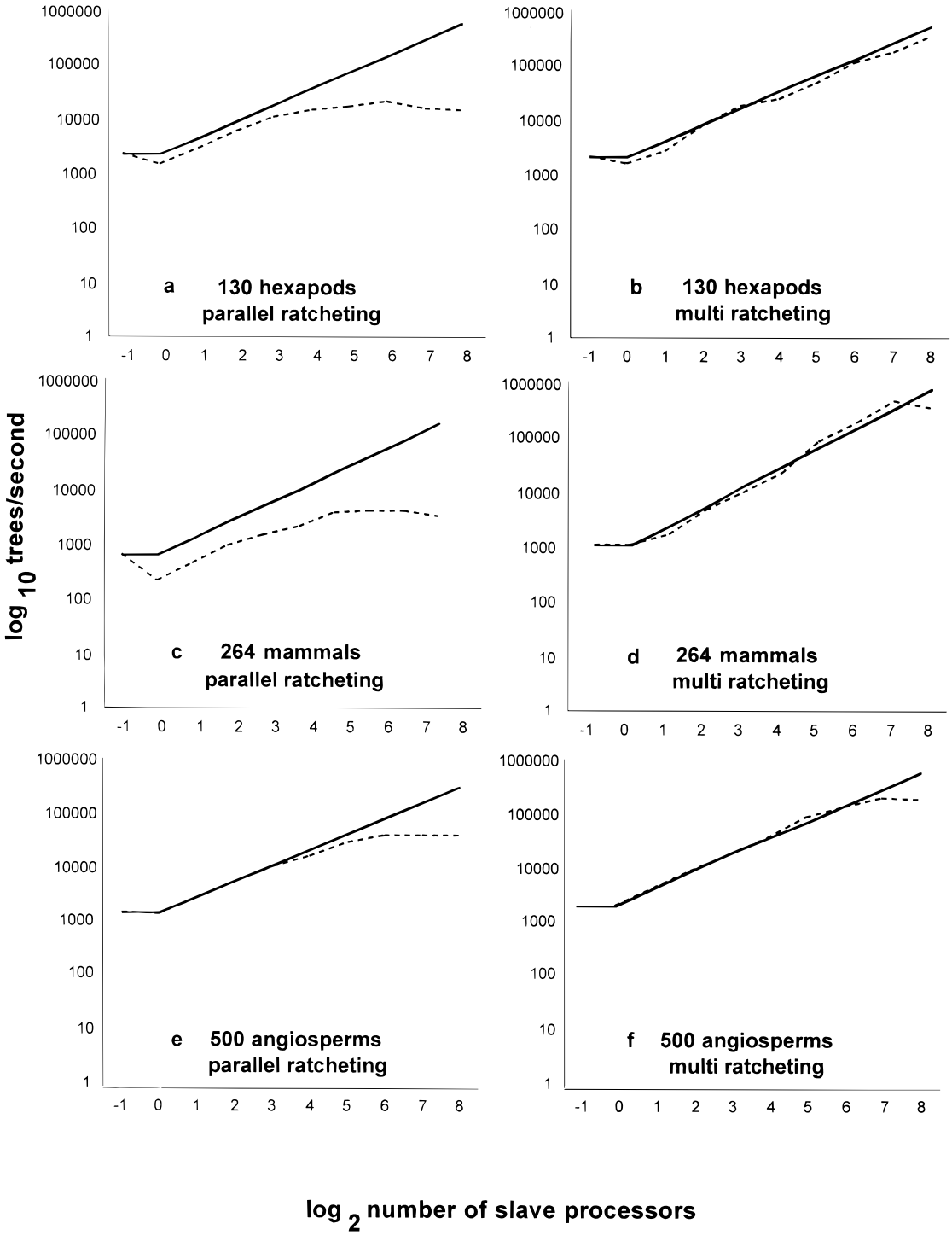

FIG. 6. Graphs of parallel efficiency of ratcheting and multiratcheting on the large cluster for three data sets: 130 hexapods, 264 mammals, and 500 angiosperms. The parallel efficiency of partitioning each ratchet job across the slave processors of the large cluster is efficient to about 16 processors. However, multiratcheting is very efficient and these results are independent of data set size.

Valuable comments on the manuscript were provided by Gonzalo Giribet, Susan Perkins, and Marc Allard. Thanks to all who provided test data sets: Todd Reeder, lizards; Mike Whiting and Jim Carpenter, hexapods; Ginny Emerson, mammals, Chase *et al.*, 1993, rbcL.

REFERENCES

Ceron, C., Dopazo, J., Zapata, E., Carazo, J., and Trelles, O. (1998). Parallel implementation of DNAml program on message-passing architectures. *Parallel Comput.* **24,** 701–716.

Cavalli-Sforza, L. L., and Edwards, A. F. W. (1967). Phylogenetic analysis: Models and estimation procedures. *Am. J. Hum. Genet.* **19,** 233–257.

Chase, M., Soltis, D., Olmstead, R., Morgan, D., Les, D., Mishler, B., Duvall, M., Price, R., Hills, H., Qiu, Y., Kron, K., Rettig, J., Conti, E., Palmer, J., Manhart, J., Sytsma, K., Michaels, H., Kress, W., Karol, K., Clark, W., Hedren, M., Gaut, B., Jansen, R., Kim, K., Wimpee, C., Smith, J., Furnier, G., Strauss, S., Xiang, Q., Plunkett, G., Soltis, P., Swensen, S., Williams, S., Gadek, P., Quinn, C., Eguiarte, L., Golenberg, E., Learn, G., Graham, S., Barrett, S., Dayanandan, S., and Albert, V. (1993). Phylogenetics of seed plants: An analysis of nucleotide sequences from the plastid gene. *Ann. MO Bot. Gard.* **60,** 528–580.

D'Haese, C. (2000). Origine de la diversité et évolution des Collemboles Poduromorphes: Phylogénies morphologiques et moléculaires. Thèse de Doctorat du Muséum National d'Histoire Naturelle, Paris.

Farris, J. S. (1978). Wagner78; program and documentation. Copyright J. S. Farris.

Farris, J. S. (1988). Hennig86 v1.5; program and documentation. Copyright J. S. Farris. Naturhistoriska riksmuseet, Stockholm, Sweden.

Farris, J. S. (1995). Parsimony Jackknifer, v.4.22. Copyright J. S. Farris. Naturhistoriska riksmuseet, Stockholm, Sweden.

Felsenstein, J. (1980–2000). PHYLIP (Phylogeny Inference Package). University of Washington. http://evolution.genetics.washington.edu/phylip.html.

Geist, A., Beguelin, J., Dongarra, W., Jiang, R., Manchek, V., and Sunderam, V. (1993). PVM 3 User's Guide and Reference Manual, Technical Report ORNL/TM-12187. Oak Ridge National Laboratory, Oak Ridge, TN.

Gladstein, D. (1997). Efficient incremental character optimization. *Cladistics* **13,** 21–26.

Gladstein, D., and Wheeler, W. (1996–2000). POY. Software for direct optimization of DNA and other data. American Museum of Natural History. New York, NY. ftp://ftp.amnh.org/pub/molecular/poy.

Goloboff, P. (1993–2000). NONA 2.0 INSUE Fundación e Instituto Miguel Lillo Miguel Lillo 205, 4000 S. M. de Tucumán, Argentina. http://www.cladistics.com/nona/nona.exe.

Goloboff, P. A. (1996). Methods for faster parsimony analysis. *Cladistics* **12,** 199–220.

Goloboff, P. A. (1999). Analyzing large data sets in reasonable times: Solutions for composite optima. *Cladistics* **15,** 415–428.

Janies, D., and Wheeler, W. (2000). POY.pdf. Documentation, sample data sets, and shell scripts for sensitivity analyses for POY, software for direct optimization of DNA and character data. ftp://ftp.amnh.org/pub/molecular/poy/poy.pdf.

Janies, D. A., and Wheeler, W. C. (2001). Theory and practice of parallel direct optimization, *In* Techniques in Molecular Systematics and Evolution. (R. Desalle, G. Giribet, and W. Wheeler, eds.) Birkhauser Verlag, Basel, in press.

Moilanen, A. (1999). Searching for most parsimonious trees with simulated evolutionary optimization. *Cladistics* **15,** 39–50.

Nixon, K. (1999). The Parsimony Ratchet, a new method for rapid parsimony analysis. *Cladistics* **15,** 407–414.

Reeder, T. (1995). Phylogenetic relationships among phrynosomatid lizards as inferred from mitochondrial ribosomal DNA sequences: Substitutional bias and information content of transitions relative to transversions. *Mol. Phylogenet. Evol.* **4,** 203–222.

Sterling, T., and Savarese, D. (1999). A coming of age for Beowulf-class computing. *In* "Euro-Par'99, LNCS 1685" (P. Amestoy *et al.*, Eds). pp. 78–88.

Sterling, T., Salmon, J., Becker, D., and Savarese, D. (1999). "How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters." MIT Press, Cambridge.

Swofford, D. (1989–1999). PAUP*4.0. Sinauer, Sunderland, MA. http://www.sinauer.com/Titles/frswofford.htm.

Wang, L., and Jiang, T. (1994). On the complexity of multiple sequence alignment. *J. Comput. Biol.* **1,** 337–348.

Wheeler, W. C. (1995). Sequence alignment, parameter sensitivity, and the phylogenetic analysis of molecular data. *Syst. Biol.* **44,** 321–331.

Wheeler, W. C., and Gladstein, D. (1994–2000). MALIGN (Multiple Alignment Using Tree Searches), version 2.7, Parallel version 1.5. American Museum of Natural History. ftp://ftp:amnh.org/pub/molecular/malign.

Wheeler, W. C., Whiting, M. F., Carpenter, J. C., and Wheeler, Q. D. (2001). The phylogeny of the insect orders. *Cladistics*, in press.