

A novel implementation and modification of the parsimony ratchet

Here we describe an alternative implementation of the parsimony ratchet (1) for use with PAUP (3). Formerly, implementations of the parsimony ratchet have been made by creating one large static batch file. We describe and present a Perl script that generates new batch files every iteration, with search instructions based on PAUP's output. We then compare the results of this script to PAUPrat (4), a program that implements the static batch file approach. Additionally, we propose a modification to the parsimony ratchet's algorithm that may help in finding parsimonious trees for very large datasets. The modified algorithm begins searches on a number of initial trees and cross-compares the scores of later trees, choosing the lowest scores as starting points for each subsequent iteration.

An alternate implementation

Under its standard implementation, the scores of subsequent trees in the parsimony ratchet (1) tend to decrease. The scores do not decrease monotonically, however (Figure 1). Given that there are occasional upswings in tree scores (Figure 1), sometimes iterations are performed on trees that do not have the lowest score found so far. These upswings are due to the standard implementation of the parsimony ratchet - a large static batch file that is read exactly as written: an initial tree is generated, and subsequent iterations are performed on the last tree created. Because random weighting ensures that no low scoring tree will get stuck on an island, an alternate methodology that warrants consideration is to always perform searches from the lowest tree score found so far. This approach raises the likelihood of monotonic decrease in subsequent tree scores. MBPR (Multi-Batch Parsimony Ratchet) is a Perl script that implements this modified ratchet algorithm. MBPR is freely available online at <http://mathbio.sas.upenn.edu/mbpr>. The script is an extension for PAUP. Instead of generating one batch file for PAUP input, MBPR generates an initial batch file, feeds it to PAUP, parses PAUP's output, and writes over the initial batch file with new instructions based on intermediate output. By having a dynamic batch file that changes based on PAUP's output, it is possible to always perform iterations from the lowest scoring tree at every stage.

Comparing the static and dynamic batch file approaches

PAUPrat is a program that implements the parsimony ratchet exactly as Nixon specified – using a static batch file (4). Performance tests contrasting the two approaches were run in a virtualized Debian operating system limited to 256 RAM with only core processes running (CPU usage <1%). The host operating system of the machine, a dual-core 64-bit AMD Athlon X2 (each 1.7 GHz), is Windows Vista. The data set used for the comparison is the 500-taxa rbcL Rice et al dataset (3) used in Kevin Nixon's paper. MBPR had identical heuristic search settings to PAUPrat, and features that PAUPrat has that MBPR does not support were kept at their default values. Each run was limited to either 20 or 45 minutes. The 20 minute runs weighted 15% of the characters randomly during iterations, and the 45 minute runs contained two runs weighted at 15% and three runs weighted at 25%. Performance was measured relative to the number of searches that successfully found one of the best three scores for the dataset (Figure 2).

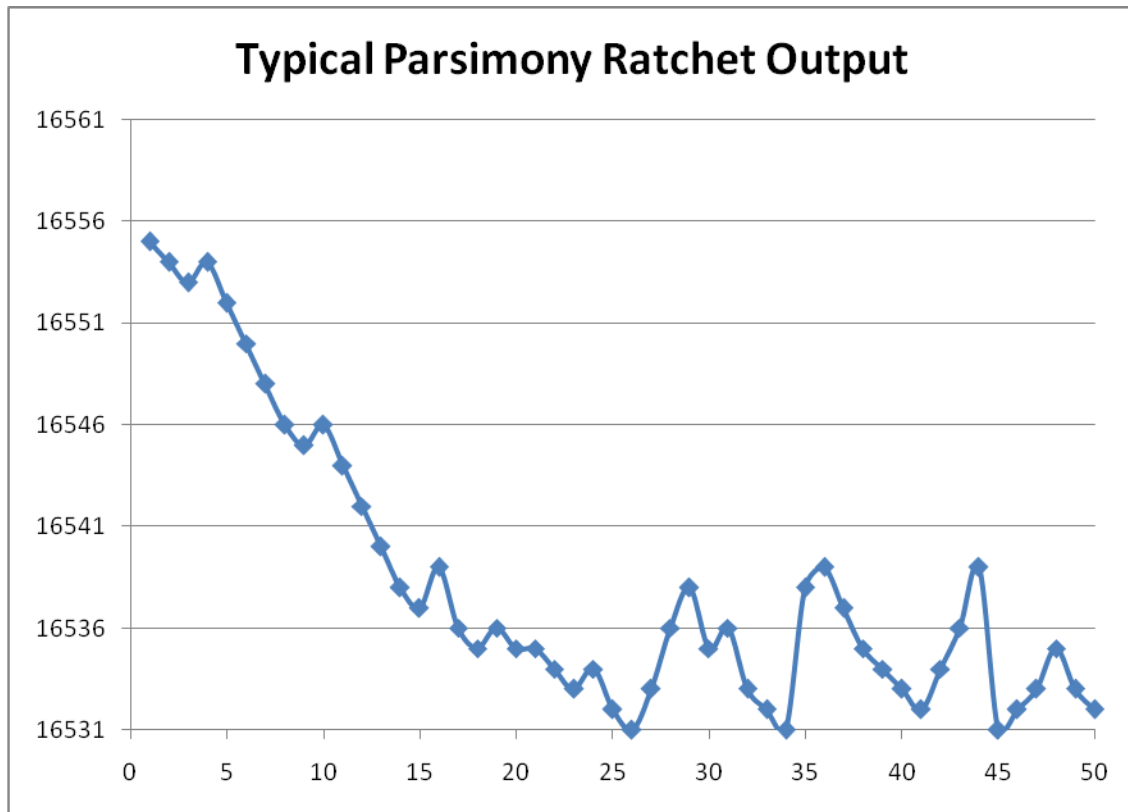
Modifying the parsimony ratchet algorithm

Using dynamic batch files it is possible to consider modifications to the parsimony ratchet's algorithm. We propose increasing the number of initial trees arbitrarily, each with a different random seed, subsequently performing iterations on each initial tree, deleting the worst scoring half of the trees, and continuing iterations on the best scoring half (see Figure 3). After each round of iterations the trees with the best scores are selected and used for subsequent iterations. Starting with a larger number of initial trees may lead to increased diversity of lowest-scoring trees with comparable computational effort.

Works

- (1) Kevin C. Nixon (1999) The Parsimony Ratchet, a New Method for Rapid Parsimony Analysis *Cladistics* 15 (4), 407-414.
- (2) Rice, K.A., M. J. Donoghue, and R. G. Olmstead. 1997. Analyzing large data sets: rbcL 500 revisited. *Syst. Biol.* 46: 554-563.
- (3) Swofford, D. L. 2003. PAUP* Phylogenetic Analysis Using Parsimony (*and other methods. Version 4. Sinauer Associates, Sunderland, Massachusetts.
- (4) Sikes, D. S. and P. O. Lewis. 2001. Beta software, version 1. PAUPrat: PAUP* implementation of the parsimony ratchet. Distributed by the authors. Department of Ecology and Evolutionary Biology, University of Connecticut, Storrs, USA.
- (5) Changing the Landscape: A New Strategy for Estimating Large Phylogenies, by Donald L. J. Quicke; Jason Taylor; Andy Purvis *Systematic Biology* © 2001 Society of Systematic Biologists

Figure 1



An example of typical parsimony ratchet output. The x axis represents iteration number and the y axis represents tree scores.

Figure 2

PAUPrat

Length	Weight	16531	16532	16533
20 Minutes	15.00%	0	3	13
20 Minutes	15.00%	0	3	26
20 Minutes	15.00%	3	7	21

MBPR

Length	Weight	16531	16532	16533
20 Minutes	15.00%	0	16	17
20 Minutes	15.00%	7	3	12
20 Minutes	15.00%	6	2	18

PAUPrat

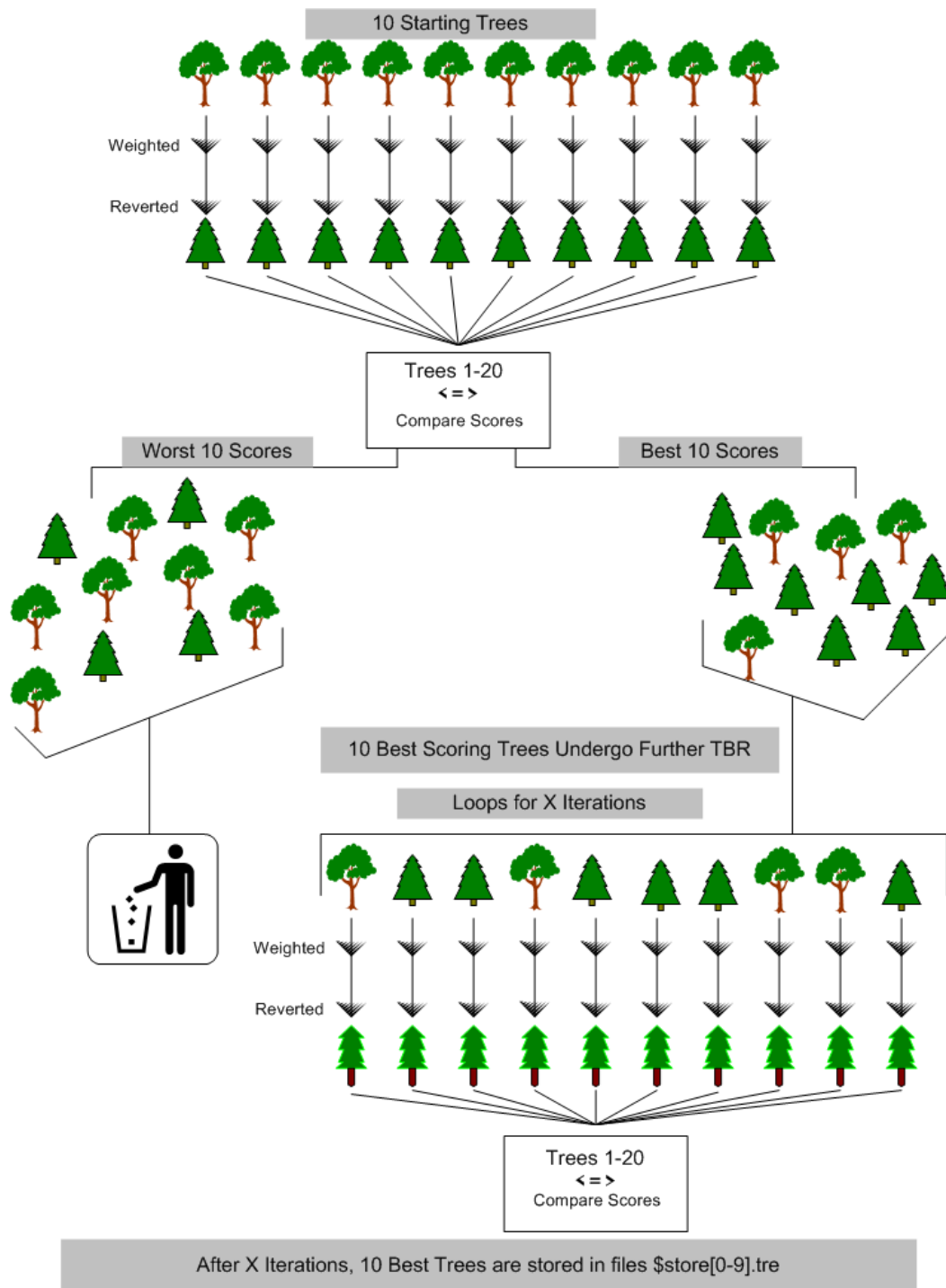
Length	Weight	16531	16532	16533
45 Minutes	15.00%	13	5	13
45 Minutes	15.00%	18	42	25
45 Minutes	25.00%	0	4	8
45 Minutes	25.00%	2	1	13
45 Minutes	25.00%	3	1	12

MBPR

Length	Weight	16531	16532	16533
45 Minutes	15.00%	93	24	15
45 Minutes	15.00%	54	58	20
45 Minutes	25.00%	61	45	24
45 Minutes	25.00%	60	34	25
45 Minutes	25.00%	15	35	10

A comparison of the performances of MBPR and PAUPrat. Each were given three 20 minute trials weighting 15% of the characters randomly during iterations, and five 45 minute trials with two trials weighting 15% of the characters randomly, and three trials weighting 25%.

Figure 3



A visual demonstration of the modified parsimony ratchet algorithm supported by MBPR.