Local Search for the Maximum Parsimony Problem

Adrien Goëffon, Jean-Michel Richer and Jin-Kao Hao

LERIA - University of Angers
2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
{goeffon,richer,hao**}@info.univ-angers.fr

Abstract. Four local search algorithms are investigated for the phylogenetic tree reconstruction problem under the Maximum Parsimony criterion. A new subtree swapping neighborhood is introduced and studied in combination with an effective array-based tree representation. Computational results are shown on a set of randomly generated benchmark instances as well as on 8 real problems (sequences of phytopathogen γ-proteobacteria) and compared with two references from the literature.

1 Introduction

Phylogeny concerns the reconstruction of the evolutionary history of a set of species identified by their nucleic acid (DNA) or amino acid (AA) sequences, also called taxa. The evolutionary relationships between species are represented by a tree, called a phylogenetic tree, whose branches reflect historical relationships. The applications of phylogeny range from classification and taxonomy to molecular epidemiology [5].

The problem of phylogeny reconstruction can be addressed using several methods. The *distance-based approach* computes a distance matrix from the taxa and tries to find a tree that approximates this matrix. Agglomerative clustering algorithms such as NJ (Neighbor-Joining) [11] and BIONJ [8] are well-known examples. The *character-based approach* searches through tree topologies to find the best tree according to an optimality criterion. The widely used Maximum Parsimony criterion [3] is such an example which states that the tree requiring the fewest number of changes (mutations) should be preferred. This *Maximum Parsimony Problem (MPP)* is known to be NP-Hard [7]. Therefore, several heuristics have been developed, including branch-swapping used in PHYLIP [4] and PAUP [12], simulated annealing [2] and other metaheuristics [1]. *Maximum Likelihood* is yet another approach for the inference of phylogeny using probabilistic estimation.

In this paper, we are interested in studying Local Search algorithms for the MPP and studying two important elements: the neighborhood relation and the internal tree representation. We evaluate a new neighborhood called Subtree Swapping Neighborhood (SSN) as well as an array-based tree representation.

** Corresponding author

2 The Maximum Parsimony Problem

Definition 1 (Phylogenetic tree). *A phylogenetic tree is a directed graph showing the relationships between a group of contemporary taxa (labels of the leaves) and their hypothetical common ancestors (internal nodes labeled by consensus sequences). If a rooted tree is used, the root is the common ancestor of all the contemporary taxa.*

Definition 2 (Consensus sequences). *Given two sequences S_1 and S_2 of length k: $S_1 =< x_1^1, x_2^1, \cdots, x_k^1 >$, $S_2 =< x_1^2, x_2^2, \cdots, x_k^2 >$ with x_i^j taken from some alphabet $\sum$, the consensus sequence S_c (parent node in a phylogenetic tree) is obtained from S_1 and S_2 by:*

$$\forall i, 1 \leq i \leq k, x_i^c = \begin{cases} x_i^1 \cup x_i^2, \text{if } x_i^1 \cap x_i^2 = \emptyset \\ x_i^1 \cap x_i^2, \text{if } x_i^1 \cap x_i^2 \neq \emptyset \end{cases}$$

The cost of the consensus sequence S_c is defined by:

$$f(S_c) = \sum_{i=1}^{k} c_i \quad \text{where} \quad c_i = \begin{cases} 1, \text{if } x_i^1 \cap x_i^2 = \emptyset \\ 0, \text{otherwise} \end{cases}$$

Definition 3 (Parsimony score of a phylogenetic tree). *Given a phylogenetic tree t and V a set of nodes whose leaves are labeled with the sequences of an initial set S, the parsimony score of t is given by :*

$$f(t) = \sum_{v \in V \setminus S} f(S_c^v)$$

where S_c^v are the consensus sequences associated to the internal nodes of t.

The goal of the Maximum Parsimony Problem is then to find a tree $t^* \in \mathcal{T}$ with the lowest parsimony score $f(t^*)$, $\mathcal{T}$ being the set of all the possible phylogenetic trees for a given set of taxa S.

3 Local Search for the Maximum Parsimony Problem

Given the NP-hardness of the MPP, local search (LS) heuristics have been massively used to find approximate phylogenetic trees. In this Section, we study four LS algorithms using a new neighborhood. First, the basic and common elements of these LS algorithms are introduced.

3.1 Tree representation and evaluation

One important issue of LS algorithms for the MPP concerns the way the trees are represented and evaluated. Here, we use an array-based representation (Fig. 1). Each node is identified by a number (N), associated with its left (L) and right (R) son, the parent (P) and the cost (C) of the node. This representation is particularly suitable for applying changes in the SSN neighborhood and convenient for computing the cost of each neighboring tree.

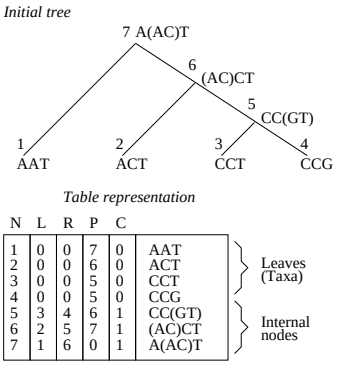

Fig. 1. Tree representation

3.2 Neighborhood

Neighborhood is a critical element of local search algorithms. The literature offers three major neighborhoods for trees: NNI (*Nearest neighborhood interchanges*) [14], SPR (*Subtree pruning and Regrafting*) and TBR (*Tree Bisection Reconnection*) [13]. NNI is a restricted neighborhood which consists in swapping two adjacent branches. SPR removes a subtree and reinserts it in other branches of the tree. TBR breaks the initial tree into two subtrees which can be reconnected to any branches of one another. It is easy to see that NNI $\subseteq$ SPR $\subseteq$ TBR.

In this study, we introduce a new neighborhood, that we call SSN (for *Subtree Swapping Neighborhood*). SSN consists in swapping two subtrees of a tree. Let $SSN_{X,Y}(t)$ be the tree obtained by exchanging the subtrees with roots X and Y of tree t such that Y (resp. X) must not be contained in the subtree rooted from X (resp. from Y). Then the SSN neighborhood $\mathcal{N}$ can be formally defined as follows $\mathcal{N} : \mathcal{T} \rightarrow 2^{\mathcal{T}}$ is such that for each $t =< N, V > \in \mathcal{T}$, a tree $t' \in \mathcal{T}$ is a neighbor of t, i.e. $t' \in \mathcal{N}(t)$, if and only if $\exists (X, Y) \in V \times V$, $SSN_{X,Y}(t) = t'$ where V is the set of nodes. As shown later, SSN, combined with our internal tree representation, contributes greatly to the efficiency of our LS algorithms.

3.3 Implemented Local Search algorithms

Pure Descent (PD) The Pure Descent (PD) algorithm accepts only better neighboring solutions. A neighboring tree t' is accepted to replace the current tree t only if $f(t') < f(t)$ (t' is more parsimonious than t). This algorithm needs no parameter and stops automatically when a local optimum (minimum) is encountered. The pure descent is very fast and may serve as a baseline reference for evaluating other algorithms.

Random Walk Descent (RWD) This algorithm combines the pure descent with the random walk strategy to accept from time to time a random neighbor (which is not necessarily better). At each iteration, with probability $p \in [0, 1]$,

a neighbor is taken randomly from the neighborhood to replace the current solution regardless of its cost; with probability $1-p$, a pure descent iteration is carried out. Here, $p = \frac{1}{\alpha.|S|^2}$, α taking values from 1 to 10 and $|S|$ being the number of species of the problem instance.

Iterative Local Search (ILS) ILS uses the pure descent to reach a first local optimum and then perturbs this local optimum by carrying out a limited number of random walks. This leads to a new solution which is then used by the pure descent to seek another local optimum. The two-steps process *Descent - Perturbation* is repeated until a predefined stop condition is met.

Simulated Annealing (SA) At each iteration, a neighbor t' is taken randomly from $\mathcal{N}(t)$ of the current tree t. t' is accepted to replace t if t' is better than t. Otherwise, t' is accepted with a probability $e^{-\frac{f(t')-f(t)}{\tau}}$ where f is the evaluation (cost) function given in Section 2 and τ is the temperature parameter which is decreased by a simple linear function. The algorithm stops when the current solution is not replaced for a fixed number of iterations.

4 Experimental Results

In this section, we compare the four LS algorithms presented above and assess their performances with respect to two references: DNAPARS of PHYLIP package [4] and LVB (both fast and slow versions) [2]. Implemented in C++, PD, RWD, ILS and SA are compiled using the `-O2` optimization option of the gcc/g++ compiler and run on Sun Fire V880 with 8 GBytes of RAM.

4.1 Benchmarks

Our benchmarks include problems having 100 to 180 sequences of a length of 100 nucleotides and were generated with *Dnatree* [10] and the Kimura two-parameter model [9] with a transition/transversion ratio fixed to 2, and an evolution rate of 0.05. We used also 8 real instances from plant pathology, composed of 69 to 95 sequences of phytopathogen γ-proteobacteria (denoted by phyto here) with 409 to 645 sites and report only here the results on one real instance since we observed very similar behavior on these instances. To run the programs, an initial tree is generated either with a random construction (Rand) or with a distance-based method (Dist). Each algorithm is run 20 to 50 times.

4.2 Comparison of PD, RWD, ILS and SA

Table 1 shows the comparative results of (PD, RWD, ILS and SA) on five classes of random instances and the phytopathogen instance, with the following information: the best cost found (f_b), the average cost (f_m), the standard deviation of the cost (σ) and the average computing time (time).

Algorithm	f_b	f_m	σ	time	Algorithm	f_b	f_m	σ	time
100.100					**160.100**				
PD	419	420,9	1,5	3m30	PD	655	658,9	2,6	13m
RWD	419	420,1	1,4	30m	RWD	655	656,6	1,9	1h20
ILS	419	419,0	0	20m	ILS	655	655,5	0,7	1h
SA	419	419,0	0	30m	SA	654	654,0	0	1h10
DNAPARS	419	419	–	4m	DNAPARS	654	654	–	65h
LVB Slow	420	420	–	>2h	LVB Slow	655	655	–	>3h
LVB Fast	421	421	–	>2h	LVB Fast	655	655	–	>3h
120.100					**180.100**				
PD	495	495,8	1,4	6m	PD	753	755,4	1,8	15m
RWD	495	495	0	40m	RWD	752	754,0	1,3	1h40
ILS	495	495,3	0,6	30m	ILS	752	753,0	1,4	1h20
SA	495	495,0	0	40m	SA	751	751,0	0	1h40
DNAPARS	495	495	–	40h	DNAPARS	751	751	–	1h20
LVB Fast	496	496	–	>1h	LVB Slow	752	752	–	>3h
LVB Slow	496	496	–	>1h	LVB Fast	752	752	–	>3h
140.100					**phyto**				
PD	683	684,6	1,2	8m	PD	731	734,8	2,6	6m
RWD	682	683,6	1,0	1h	RWD	730	731,0	1,1	40m
ILS	683	684,2	1,1	40m	ILS	731	732,8	1,5	30m
SA	682	682,0	0	50m	SA	729	729,8	0,7	40m
DNAPARS	682	682	–	51h	DNAPARS	731	731	–	14h
LVB Slow	683	683	–	>5h	LVB Slow	764	764	–	>4h
LVB Fast	685	683	–	>4h	LVB Fast	740	740	–	>4h

Table 1. Comparison of PD, ILS, RWD, SA, DNAPARS and LVB

From Table 1, one observes that PD is able to find good solutions with very short computation times compared with other algorithms. RWD finds a little better solutions, but needs more computation time. We suspect that executing RWD more times may lead to even better solutions. ILS, even with a long computation time, is not competitive. This is somewhat unexpected given that it uses a perturbation techniques to re-start PD. One possible explanation would be that the simple re-start technique used by PD (recall that PD was run 5 times) is more appropriate than re-starting PD with a solution near a local optimum. Finally, SA is the most powerful algorithm, able to find the most parsimonious trees with reasonable computation times.

4.3 Comparisons of LS algorithms with LVB and DNAPARS

From Table 1, one observes first that in terms of solution quality, SA and DNAPARS find the same results for random instances, and SA finds better solutions for the real instance. However, SA is much faster than DNAPARS to find solutions of the same quality. This is particularly true when the problem instance is of larger size. Indeed for still larger instances (with more than 200 sequences, not reported here), DNAPARS did not finish after 2 days of computation while SA needs 1 to 2 hours to obtain near-optimal solutions. For the phytopathogen instance, our SA algorithm obtains better result than DNAPARS (with a cost of 729 against 731). If we consider the results of LVB, one observes easily that both the fast and slow versions of LVB are often dominated by our algorithms, both in terms of solution quality and computation time.

5 Conclusion

An empirical study of four local search algorithms is carried out for the phylogenetic tree reconstruction with the Maximum Parsimony criterion. These algorithms are tested on both random instances and real problems. They are also compared with two references from the literature, showing competitive results. This study confirms that local search remains a very promising approach for the Maximum Parsimony Problem. This study has allowed us to assess the proposed SSN neighborhood and the array-based tree representation. Based on the results, we are investigating an improved local search algorithm using an evolutionary SSN neighborhood combined with a noisy evaluation function. Experimental validations are on the way by using very large instances (up to 500 taxa and 2 000 sites, including the *Zilla* data set).

Acknowledgment This work is partially supported by the French Ouest Genopole®. We thank the Plant Pathology Lab. from the INRA of Angers for providing us with the phytopathogen sequences.

References

1. A.A. Andreatta and C.C. Ribeiro.Heuristics for the phylogeny problem. *Journal of Heuristics* 8:429-447, 2002.
2. D. Barker. LVB: parsimony and simulated annealing in the search for phylogenetic trees. *Bioinformatics* 20:274-275, 2003.
3. J.H. Camin and R.R. Sokal. A method for deucing branching sequences in phylogeny. *Evolution* 19:311-326, 1965.
4. J. Felsenstein. *Phylogenetic Inference Package (PHYLIP)*, 1993.
5. J. Felsenstein. Inferring Phylogenies. *Sinauer*, 2003.
6. W. Fitch. Towards defining course of evolution: minimum change for a specified tree topology. *Systematic Zoology* 20:406-416, 1971.
7. L.R. Foulds and R.L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics* 3:43-49, 1982.
8. O. Gascuel. BIONJ: An improved version of the NJ algorithm based on a simple model of sequence data. *Molecular Biology and Evolution* 14:685-695, 1997.
9. M. Kimura. A simple model for estimating evolutionary rates of base of base substitutions through comparative studies of nucleotide sequence. *Journal of Molecular Evolution* 16:111-120, 1980.
10. M.K. Kuhner and J. Felsenstein. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, 11:459-468, 1994 (Erratum 12:525, 1995).
11. N. Saitou and M. Nei. Neighbor-joining method : A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* 4:406-425, 1987.
12. D.L. Swofford. *PAUP: Phylogenetic analysis using parsimony.* Sinauer Associates, 1998.
13. D.L. Swofford and G.J. Olsen. in D.M. Hillis and C. Moritz (Ed.) Phylogeny Reconstruction. *Molecular Systematics*, chapter 11:411-501, 1990.
14. M.S. Waterman and T.F. Smith. On the similarity of dendograms. *Journal of Theoretical Biology* 73:789-800, 1978.