LERIA
Faculté des Sciences
2 Boulevard Lavoisier
49045 Angers Cedex 01

Technical Report:
Three new techniques to improve phylogenetic reconstruction with Maximum Parsimony

Research

Version 1.0
April 28, 2008

Jean-Michel Richer
Office H206
Tel : 02-41-73-52-34
Email : jean-michel.richer@univ-angers.fr

28 April 2008

Abstract

The Maximum Parsimony problem aims at reconstructing a phylogenetic tree from DNA, RNA or protein sequences while minimizing the number of evolutionary changes. Much work has been devoted by the Computer Science community to solve this NP-complete problem and many techniques have been used or designed in order to decrease the computation time necessary to obtain an acceptable solution. In this paper we report on three new improvements (implemented in the software Hydra) that can lead to shorten the time to solve the MP problem and get a solution of better quality. Those techniques can be integrated without too much effort to existing softwares.

1 Introduction

Phylogenetics, also kown as Phylogenetic Systematics, is the formal name for the field within Biology that reconstructs evolutionary history of species and studies the patterns of relationships among organisms. Phylogenetics studies the connections between groups of species (as understood by an ancestor / descendant relationships) which are represented by a phylogenetic tree, whose leaves represent contemporary or extinct species and internal nodes correspond to hypothetical ancestors.

Recent advances in genomics, including whole-genome sequencing have mainly influenced phylogenetics. In the past, morphological characters (like size, color, number of legs, etc...) were used for inferring phylogenies. With today's data obtained from the sequencing programs, we can compare organisms from the information extracted from their genetic material (DNA, RNA) or their proteome (protein sequences). This is particularly useful when dealing with prokaryote organisms like bacteria or viruses for which morphological characters are difficult to define or identify. *Molecular* phylogenetics has then become an important field in Bioinformatics and finds many applications in biology and medicine like genetic evolution, taxonomy and classification, or virus detection and mutation [15].

Much work has been devoted to the problem of phylogeny reconstruction since the 1960s. The general principle behind phylogenetic methods is to find a tree that minimizes sequence changes or mutations. We can bring three different approaches to light:

- *Distance methods*, inspired by the clustering work of Sokal and Sneath [24], introduced in 1967 by [3] or by [7] rely on a matrix of distances observed between species. The most well-known algorithm of this class of methods is the *Neighbor-Joining* from [22], improved by [9]. Those methods are very efficient for they rely on an algorithm of polynomial complexity but they sometimes lack robustness.

- *Probabilistic methods* are based on a model of evolution of characters. The Maximum Likelihood (ML), introduced by [5] in 1981 provides a general framework that consists in inferring the most probable phylogeny that maximizes the likelihood of observed sequences. Although ML is popular for phylogenetic inference because it is considered as a robust method, it is more computationally expensive than other methods.

- *Cladistic methods* are based on a matrix of given characters. The most well-known method of this class relies on the Maximum Parsimony (MP) criterion [4] as it is quite simple to apprehend. Such a method aims at building a binary tree that minimizes the number of changes without resorting to a particular model of evolution. The cost of a tree can be computed in polynomial time [6]. However the search for an optimal tree is NP-complete as shown by [8]. In theory, this problem can be solved by enumerating all possible topologies of a binary tree and retaining the topologies which have the smallest cost. As there is an exponential number of topologies, such an exhaustive search is only viable for very small instances. This is why heuristics methods have been applied to this problem in order to obtain a near optimal tree with reasonable computation time [13, 19].

In this article we present three new improvements to tackle phylogeny reconstruction with Maximum Parsimony (MP) under Fitch's criterion. The first improvement is based on the VNS (Variable Neighborhood Search) principle and aims at reducing the number of iterations to quickly attain a *good* solution when using Local Search algorithms. On real instances this technique leads to interesting results compared to TNT [12], the most efficient software to solve the MP problem.

The second improvement concerns the definition of a new crossover operator for a Genetic Algorithm designed to solve the MP problem. This crossover called DiBIP (for Distance-Based Information Preservation) aims to preserve representative properties of parents in terms of *topological distance* between sequences. It can be considered as an alternative to traditional transformation of trees.

The third improvement takes into account the ability of modern x86 processors (Intel, AMD) to vectorize treatments in order to efficiently compute Fitch's score and the hypothetical sequence of each internal node. This technique which to our knowledge has never been described, helps decrease the computation time by 80 % on all x86 modern processors.

The rest of the paper is organized as follows. In the next section we formally introduce the MP problem. We then describe each of the three techniques previously mentioned. An experimental section is devoted to the verification of the influence of those techniques implemented in the software Hydra compared to the

software TNT. We finally conclude that those techniques are quite efficient and could improve the efficiency of MP softwares.

2 Phylogeny reconstruction and Maximum Parsimony

Phylogeny generally takes as input a multiple alignment which is a matrix of characters composed of n lines (related to a set S of species, where $|S| = n$) and k columns which represent the characters of the sequences. Each sequence is also called a taxon (or taxonomic unit, plural taxa). Each character of the matrix belongs to an alphabet Σ. The aim of the Maximum Parsimony problem is to find a phylogenetic tree (i.e. generally a binary tree, rooted or unrooted) that minimizes the number of changes (or mutations) between sequences. Each leaf of the tree is associated to one of the n species and the cost (or number of mutations) of the overall tree can be estimated by building hypothetical *sequences of parsimony* from the leaves toward the root of the tree. More precisely we can formulate the following definitions:

Definition 1 (Sequence of parsimony - see [6]) *Given two sequences S_1 and S_2 of length k characters such that $S_1 = x_1 \cdots x_k$, $S_2 = y_1 \cdots y_k$ with $\forall i \in \{1..k\}, x_i, y_i$ belong to the power set $\mathcal{P}(\Sigma)$, the sequence of parsimony of S_1 and S_2, noted $F(S_1, S_2) = z_1 \cdots z_k$ is obtained by :*

$$\forall i, 1 \leq i \leq k, z_i = \begin{cases} x_i \cup y_i, \text{ if } x_i \cap y_i = \emptyset \\ x_i \cap y_i, \text{ otherwise} \end{cases}$$

The cost (or number of mutations) of a sequence of parsimony is defined by:

$$\phi(F(S_1, S_2)) = \sum_{i=1}^{k} c_i \quad \text{where} \quad c_i = \begin{cases} 1, \text{if } x_i \cap y_i = \emptyset \\ 0, \text{otherwise} \end{cases}$$

Definition 2 (Rooted Binary Tree of Parsimony) *Let S be a set of n aligned sequences of length k where each character of the sequence is expressed over a given alphabet Σ. Let $T = (V, E)$ be a binary tree, where $V = \{v_1, \ldots, v_r\}$ is the set of nodes and $E \subseteq \{(u, v)/u, v \in V\}$ is the set of edges. T is called a binary tree of parsimony of S if :*

- *there exist $r = 2 \times n - 1$ nodes partitioned in two subsets:*
  - *I : a set of internal nodes composed of $n - 1$ nodes each having 2 descendants,*
  - *L : a set of leaves composed of n nodes with no descendant.*
- *there exists a bijection from the set of sequences S to the set of leaves L,*
- *each internal node w of I is assigned to a (hypothetical) sequence of parsimony $S_w = F(S_u, S_v)$.*

Definition 3 (Cost of a Tree of Parsimony) *Let T be a binary tree of parsimony of a set of sequences S. The cost (or score) of T, $\phi(T)$ is equal to $\sum \phi(S_w), \forall w \in I$.*

Definition 4 (Maximum Parsimony Problem) *Given a set S of n sequences of length k, expressed over an alphabet Σ, find the most parsimonious tree T of S such that the score of parsimony of T is minimum.*

For a set of S of sequences, there are $\prod_{i=3}^{|S|}(2i-3))$ possible parsimony trees. The MP problem is thus a highly combinatorial search problem.

3 New techniques

In this section we describe three techniques that helped us improve the efficiency of the resolution of the Maximum Parsimony problem under Fitch's parsimony. Those techniques have been implemented in the software Hydra[1] (see section Availability).

[1] for HYbrid Descent Recombination Algorithm

3.1 Intelligent variable neighborhood search

The most powerful softwares based on Maximum Parsimony use local search algorithms [12]. Local Search (LS) is a class of metaheuristics for solving computationally hard optimization problems [17].

The principle of LS, applied to phylogenetic reconstruction, is to start from an initial tree, randomly generated or built with a greedy algorithm, that is then iteratively improved. The LS algorithm used in Hydra is a *first improvement* descent which can be stated as follows: while there is a tree t' in the neighborhood of t whose score is better or equal than the score of t then replace t by t' and continue the iteration process during a given number a iterations.

This process can be used with one of the three well-known neighborhoods traditionally used by MP softwares: NNI *Nearest Neighbor Interchange* [26], SPR *Subtree Pruning Regrafting* [25], TBR *Tree-Bisection-Reconnection* [25].

NNI consists in swapping two adjacent branches of the tree. We can say that NNI is a *small* size neighborhood for which a tree of n leaves has $(2n-6)$ neighbors [21].

SPR is a strategy which cuts a branch and reinserts it elsewhere. There are $2(n-3)(2n-7)$ possible SPR rearrangements [1] for each tree which makes it a *medium* size neighborhood.

TBR is an extension of SPR and consists in breaking the tree in two subtrees which will be reconnected from one of their branches. Given a tree, the *large* TBR neighborhood induces at most $(2n-3)(n-3)^2$ neighbors [1].

Finally the following property can be verified [18]: NNI $\subseteq$ SPR $\subseteq$ TBR.

An interesting improvement consists in succesively using a set of neighborhood relations. For example $\{$NNI, STEP, SPR$\}$ or $\{$SPR, 2-SPR, ..., l-SPR$\}$, where l-SPR is the neighborhood relation defined by l SPR moves and STEP is a modified version of SPR for which only leaves are pruned. This can be considered as an application of the VNS (Variable Neighborhood Search) metaheuristic, proposed by [14] to the MP problem. In [2, 20] authors reported very good solutions for a set of benchmarks.

In [11], we have introduced a new neighborhood called *Progressive Neighborhood* (PN). The advantage of PN is that it minimizes the number of iterations used to find a tree of minimum score. Similar to the VNS heuristic, PN modifies the size of the neighborhood during the search. But, contrary to VNS, we start from a medium (or large) size neighborhood like SPR and progressively reduce it to the small size NNI neighborhood using a parametric neighborhood relation that evolves during the search. In order for the neighborhood to evolve, we have defined in [10] a topological distance on trees that enables us to build a distance matrix for a set of taxa given a tree topology and also control the size of the search space.

Definition 5 (Topological distance) *Let i and j be two taxa of a tree T. We define $\delta_T(i,j)$ as the topological distance between i and j as the number of edges of the path between parents of i and j, minus 1 if the path contains the root of the tree.*

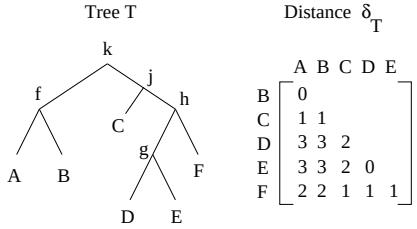

Figure 1: Example of topological distance δ_T

For example, on figure 1, A and B have the same parent f, so $\delta_T(A,B) = 0$, and $\delta_T(A,D) = 3$ because the number of edges between f and g is 4 ($f \to k \to j \to h \to g$), and as we pass through the root node k, we decrease the value of one unit. Note that for the topological distance, we consider trees as unrooted, this is why we remove one unit when passing through the root node.

The reduction scheme used in Hydra takes into account a parameter M which corresponds to a maximum number of LS iterations. Another parameter d controls the size of the neighborhood and is defined as the distance between a pruned edge and the edge where it is reinserted (i.e. distance δ between their two descendant nodes). We use a linear reduction scheme for d [11].

Experiments carried out proved the efficiency of this technique compared to other neighborhoods like NNI, SPR and TBR. In particular, we showed that important tree transformations must be performed at early stages of the search using a medium or large size neighborhood like SPR or TBR and that only minor refinements are necessary at the end of the LS process where a neighborhood like NNI is sufficent.

3.2 A crossover operator based on distances

Genetic Algorithms (GA) mimic the process of evolution of species [16]. They are time consuming but can lead to results of better quality than other metaheuristics methods. Traditionally, GA work with a constant number of individuals (population); at each step of the algorithm two individuals are selected and are bred using a crossover operator to obtain a set of one or two (sometimes much more) new individuals (children). Then, some slight modifications on the children are eventually performed (mutations) and the selection process keeps or rejects them (for example we can keep the children that are better than their parents).

The software Hydra is an hybrid algorithm based on a GA and its general scheme is given on Algorithm 1. The function *ChooseParents* operates with a tournament selection strategy. Two groups of 20% of the individuals are constituted. Two individuals that represent the best individuals of each group are selected. The *LS+PN* function is a LS Descent using a Progressive Neighborhood, it was designed to improve a child and is part of the mutation process. The *Replace* function, which replaces one individual by the child tree T obtained from the crossover operator DiBIP, the replacement is effective if T is not already present in the population P. The individual that will be removed from P can be the oldest or the closest to T.

Algorithm 1 Hybrid Genetic and Local Search algorithm (Hydra) for the MP problem

input: A : an alignment of taxa, N : the size of the GA population, M number of LS iterations
output: The most parsimonious tree found

P = *GeneratePopulation(A,N)*
for a given number of crossovers **do**
  $(T_1, T_2) \leftarrow$ *ChooseParents(P)*
  $T \leftarrow$ *DiBIP(T_1, T_2) // generate new tree*
  $T \leftarrow$ *LS+PN(T,M) // mutation process*
  $P \leftarrow$ *Replace(P,T) // replacement*
end for
return the best tree found

In the case of phylogeny reconstruction the individuals are trees and breeding them can sometimes become a complex operation. Traditionally, topological crossovers on trees follow the *subtree cutting and regrafting* strategy which consists in removing a subtree T_1^* from a parent T_1 and reinsert it in the second parent T_2. Duplicated sequences (leaves) are removed from T_2.

This process suffers a major drawback. The topological information of trees, which are responsible of their parsimony score, is not entirely used to improve the tree. In that sense we can consider that those crossovers perform a *blind* search because they are based on partial information and arbitrary choices. It would then be necessary to be able to pass in review millions of trees per second to keep up with the efficiency.

On the contrary, we propose to use meaningful topological information and pass it to the offspring. The aim is to avoid useless computing and bring the search to potentially interesting areas. We use the topological distance δ (see Definition 5) between leaves of the trees which enables us to build a distance matrix of sequences given a tree topology.

The crossover operator DiBIP [10] takes into account two trees and computes their topological distance matrices M_1 and M_2 (see Figure 1). We then combine the two matrices by calculting $M_3 = \alpha M_1 + \beta M_2$ where $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$. Generally we will take $\alpha = \beta = 0.5$. From M_3, using the same principle as the UPGMA method [23], we build a new tree. The key idea is the preservation of semantic information shared by the parents: if two sequences are topologically close (or far) in both parent trees, then this property should be conserved for the child. Moreover, this crossover enables to diversify the search: the child is fully rebuilt from these information.

We have performed some tests in [10] on real and randomly generated instances and have confronted our algorithm to the GRASP+VND [20] method which is an evolutionary algorithm which can be considered

close to Hydra. On random instances from [2, 20], we could outperform the results of 19 instances out of 20 compared to GRASP+VND.

3.3 Software improvement using SSE2 instructions

Version 1.1 of Hydra was improved compared to version 1.0, using SSE2 (Streaming SIMD Extensions) instructions of modern x86 processors (Intel, AMD). SSE2 instructions help vectorize the code, i.e. apply the same instruction on multiple data at the same time consequently reducing the overall execution time. The main function that benefits from the use of vectorization is the computation of a parsimony sequence (see definition 1) which is implemented figure 2 (in C) and takes as input two taxa `t1` and `t2`. The output is the hypothetical taxon `t3` and the number of changes returned by the function.

```
int fitch(char t1[], char t2[], char t3[]) {
  int changes=0;
  for (int i=0;i<length;++i) {
    t3[i]= t1[i] & t2[i];
    if (t3[i]==0) {
      ++changes;
      t3[i]= t1[i] | t2[i];
    }
  }
  return changes;
}
```

Figure 2: Parsimony function traduction of Definition 1

The vectorization of the function is not straightforward but gives a 90% improvement on Intel Core 2 Duo processors, while other architectures (pentiumII/III/4, pentium-M, Athlon 64, Sempron) provide a 70 to 80 % improvement. This improvement enabled us to divide by a factor of 4 the computation time of Hydra 1.0. This function was written in assembler using some of the 8 XMM registers of the SSE units in order to treat 16 bytes at a time. The source file was compiled using the assembler *nasm* under Linux. The alphabet of the DNA sequences is generally composed of 6 different symbols : $-, A, C, G, T, ?$. Where $-$ represents a gap and $?$ an undefined character. In order to efficiently perform the union and intersection of definition 1, each character is represented by a power of 2, from $2^0 = 1$ ($-$) to $2^4 = 16$ (T), except for $?$ which can represent any other character and is then coded by the value $31 = 1 + 2 + \cdots + 16$. Union can be performed by the binary-OR (`|`) and intersection by the binary-AND (`&`). Thus in the case of DNA (or RNA), the characters can be represented using one byte and a SSE register (of size 128 bits) can hold and apply operations on 16 bytes at a time. We can then decompose each XMM register as 16 different cells from $XMM[0]$ to $XMM[15]$.

The principle of the calculation is the following : we first load into registers $XMM0$ and $XMM1$ the first 16 bytes of each taxon (`t1` and `t2`). In $XMM2$ and $XMM3$, we respectively compute the binary-AND and the binary-OR of $XMM0$ and $XMM1$ using instructions PAND and POR (for parallel AND and parallel OR). Then, we compare $XMM2$ with a vector of zero using the instruction PCMPEQB (which performs a parallel comparison of each byte of two XMM registers) in order to determine which bytes of $XMM3$ will replace the zero values of $XMM2$. The result is stored in register $XMM5$. The result of PCMPEQB is that $XMM5[i] = 0$, if originally $XMM2[i] = 0$ otherwise $XMM5[i] = 255$. As a consequence, we can use $XMM5$ and combine it with $XMM2$ and $XMM3$ to get the final result of taxon `t3` by calculating : $(\mathrm{XMM5} \,\&\, \mathrm{XMM3}) | (NOT(\mathrm{XMM5}) \,\&\, \mathrm{XMM2})$. This process is repeated every 16 bytes until we reach the last bytes of the taxa. When the size of the taxa is not a multiple of 16, the last part of the taxon `t3` is computed using a traditional implementation which treats one byte at a time. The number of changes is evaluated using an efficient version of the function POPCOUNT (for POPulation COUNT) which counts the number of bits set to 1 in a general purpose register. We refer the reader to the source code for more details (see section Availability). Note that the POPCOUNT function is not natively implemented on processors but is part of the new SSE4.2 instructions set of the next generation of Intel Core 2 processors (the Nehalem family of processors which should be available in 2008). We expect to

obtain a 5 % improvement by replacing the current POPCOUNT function by its native SSE4.1 version. An overview of the assembler code is given figure 3, where $XMM7$ is a vector of zero.

```
movdqa xmm0,[esi] ; load t1 into xmm0
movdqa xmm1,[edi] ; load t2 into xmm1

movdqa xmm2,xmm0 ; xmm2 <- xmm0 & xmm1
pand   xmm2,xmm1

movdqa xmm3,xmm0 ; xmm3 <- xmm0 | xmm1
por    xmm3,xmm1

movdqa xmm5,xmm2 ; compare xmm2 to 0
pcmpeq xmm5,xmm7 ; and store in xmm5

movdqa xmm4,xmm5 ; compute ~xmm5
pxor   xmm4,xmm7 ; in xmm4

pand   xmm3,xmm4 ; xmm3 <- xmm3 & ~xmm5
pand   xmm2,xmm5 ; xmm2 <- xmm2 & xmm5
por    xmm2,xmm3 ; xmm2 <- xmm2 | xmm3

movdqa [ebx],xmm2 ; store result in t3
```

Figure 3: Traduction of the parsimony function in x86 assembler using SSE instructions.

4 Benchmarks and results

Hydra was compared to the most powerful software for phylogenetic reconstruction called TNT [12] (Tree analysis using New Technology) which is based on LS and implements a set of efficient techniques such as tree drifting, parsimony ratchet and sectorial search. TNT was used with default parameters specified in the documentation.

We have chosen a set of real instances from the TreeBase site (www.treebase.org). TreeBase is a relational database of phylogenetic information which stores phylogenetic trees and the data matrices used to generate them from published research papers. Only instances that had more than 100 taxa were chosen and their characteristics (number of taxa, length of taxa) are reported table 1. Results are reported table 2 where *# iter* represents the number of LS iterations performed (in millions) and *score* stands for Fitch's score. A lower score means a better score.

problem	# taxa	length	problem	# taxa	length
m0808	178	3453	m2123	198	1426
m0972	155	355	m2725	210	8245
m1038	297	2021	m2780	119	2020
m1902	209	977	m3275	117	5098
m1987	134	618	m3452	116	1157
m2055	299	2064	m3453	137	995

Table 1: Characteristics of benchmarks from TreeBase.

The results show that Hydra obtains results of equal or better quality than TNT with less iterations except for problem m2725. For example, for the problem *m0808*, we could find a solution of value 23,777 in 18×10^6 iterations while TNT obtains a solution of value 23,782 in 40×10^6 iterations. For *m2780*, we could get a better cost of 11,488 but using more iterations.

problem	Hydra		TNT	
	# iter	score	# iter	score
m0808	18	**23,777**	40	23,782
m0972	4.5	**1,529**	14	1,532
m1038	4.5	**12,369**	99	12,376
m1902	4.5	**6,125**	44	6,131
m1987	18	**2,073**	30	**2,073**
m2055	18	**2,597**	47	2,598
m2123	4.5	**1,925**	46	**1,925**
m2725	28	157,477	63	**157,093**
m2780	18	**11,488**	**11**	11,490
m3275	**9**	**21,935**	**9**	**21,935**
m3452	4.5	**3,602**	23	**3,602**
m3453	4.5	**3,904**	24	**3,904**

Table 2: Comparison between Hydra and TNT.

The number of iterations was fixed according to the number of taxa and the length of taxa for each problem. All problems used a constant population of 40 individuals and M, the number of LS iterations was set to $200,000$. For problem *m0808*, we perform 50 crossovers and each LS process performs M iterations. The total number of iterations is then $(40+50) \times 200,000 = 18 \times 10^6$. Indeed, the initial trees, once generated are improved using LS+PN.

As mentioned previously, only problem *m2725* could not efficiently be solved by Hydra. We are currently investigating and we have written a tree viewer and editor in Java to find out why there is such a difference between TNT and Hydra. This difference can be explained by different tree topologies, but until now we couldn't figure out why the topologies are so distant. *m2725* is probably what we could call a hard instance as TNT uses many TBR searches and get stuck at the value 157,113 during 42 millions iterations, that is 2/3 of the total number of iterations.

Finally, note that the computation time of TNT is less important than Hydra because TNT is a LS algorithm and implements optimization techniques in order not to recompute the overall tree when looking for a neighbor of lower cost. Such technique is not yet implemented in Hydra and will be introduced in its next release.

5 Conclusion

In this article we have introduced three new techniques to efficiently solve the Maximum Parsimony problem. The first one aims to decrease the number of local search iterations in order to reduce the computation time of the search process. The second, introduces a new crossover operator for trees which is designed to take into account topological properties of trees and is used by a Genetic Algorithm. The crossover DiBIP has the ability to diversify the search. The last one, is an implementation improvement which relies on the capabilities of modern processors to vectorize data treatments. All those techniques are implemented in the software Hydra and have proven to provide good quality results compared to the state of the art program TNT. We think that it would be worth to implement those techniques into existing MP softwares. The next step in improving Hydra will be to implement optimization techniques for the LS mutation process as implemented in TNT in order to significantly decrease the computation time.

6 Availability:

HYDRA is distributed with C++ source code, benchmarks and documentation and is freely available at the following URL : http://www.info.univ-angers.fr/pub/richer/rec.php. It should run under all Unix/Linux platforms. There is also a binary for Windows platforms.

References

[1] B.L. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5(1):1–15, 2000.

[2] A.A. Andreatta and C.C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics*, 8:429–447, 2002.

[3] L.L. Cavalli-Sforza and A.W.F. Edwards. Phylogenetic analysis: models and estimation procedures. *Evolution*, 32:550–570, 1967.

[4] A.W.F Edwards and L.L. Cavalli-Sforza. The reconstruction of evolution. *Annals of Human Genetics*, 27:105–106, 1963.

[5] J. Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.

[6] W. Fitch. Towards defining course of evolution: minimum change for a specified tree topology. *Systematic Zoology*, 20:406–416, 1971.

[7] W.M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155(3760):279–284, 1967.

[8] L.R. Foulds and R.L. Graham. The steiner problem in phylogeny is np-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

[9] O. Gascuel. On the optimization principle in phylogenetic analysis and the minimum evolution criterion. *Biology and Evolution*, 17:401–405, 2000.

[10] A. Goëffon, J.M. Richer, and J.K. Hao. A distance-based information preservation tree crossover for the maximum parsimony problem. *Lecture Notes in Computer Science*, pages 761–770, 2006.

[11] A. Goëffon, J.M. Richer, and J.K. Hao. Progressive tree neighborhood applied to the maximum parsimony problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2007.

[12] P. A. Goloboff, J. S. Farris, and K. Nixon. Tnt : Tree analysis using new technology. http://www.cladistics.com/aboutTNT.html, 2003.

[13] P.A. Goloboff. Character optimisation and calculation of tree lengths. *Cladistics*, 9:433–436, 1993.

[14] P. Hansen and N. Mladenovic. *Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization*, chapter An introduction to Variable neighborhood search, pages 433–458. Kluwer Academic Publishers, Dordrecht, edited by s. voss et al. edition, 1999.

[15] D.M. Hillis, C. Moritz, and B.K. Mable. *Molecular Systematics*. Sinauer Associates, Inc., 1996.

[16] J.H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, 1975.

[17] H.H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2004.

[18] D. R. Maddison. The discovery and importance of multiple islands of most-parsimonious trees. *Systematic Zoology*, 43(3):315–328, 1991.

[19] K.C. Nixon. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15:407–414, 1999.

[20] C.C. Ribeiro and D.S. Vianna. A grasp/vnd heuristic for the phylogeny problem using a new neighborhood structure. *International Transactions in Operational Research*, 12:1–14, 2005.

[21] D.F. Robinson. Comparison of labeled trees with valency three. *J. Combin. Theory*, 11:105–119, 1971.

[22] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.

[23] R.R. Sokal and C.D. Michener. A statistical method for evaluating systemaic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.

[24] R.R. Sokal and P.H.A. Sneath. *Principles of Numerical Taxonomy*. W.H. Freeman, San Francisco, 1963.

[25] Swofford and Olsen. *Molecular Systematics*. D.M. Hillis and C. Moritz, 1990.

[26] M.S. Waterman and T.F. Smith. On the similarity of dendograms. *Journal of Theoretical Biology*, 73:789–800, 1978.