A Memetic Algorithm for Phylogenetic Reconstruction with Maximum Parsimony

Jean-Michel Richer[1] and Adrien Goëffon[2] and Jin-Kao Hao[1]

[1] University of Angers, LERIA, 2 Bd Lavoisier, 49045 Anger Cedex 01, France
[2] LaBRI (UMR 5800) Université de Bordeaux 351 cours de la Libération 33405 Talence Cedex, France

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 research community to solve this NP-complete problem and many algorithms and techniques have been devised in order to find high quality solution with reasonable computational resources. In this paper we present a memetic algorithm (implemented in the software Hydra) which is based on an integration of an effective local search operator with a specific topological tree crossover operator. We report computational results on a set of 12 benchmark instances from the literature and demonstrate its effectiveness with respect to one of the most powerful software (TNT). We also study the behavior of the algorithm with respect to some fundamental ingredients.

Keywords: Maximum Parsimony, phylogeny, progressive descent, tree crossover, memetic algorithm.

1 Introduction

Phylogenetics, also known as Phylogenetic Systematics, is the formal name for the field within Biology that reconstructs evolutionary history of species. Phylogenetics studies the connections between groups of species (as understood by an ancestor / descendant relationships) which are represented by a phylogenetic tree. In such a phylogenetic tree, the 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 [13].

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 [19], introduced in 1967 by [2] or by [6] rely on a matrix of distances observed between species. The most popular algorithm of this class of methods is probably the *Neighbor-Joining* from [17], improved by [8]. 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 [4] 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 [3] 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 [5]. However, the search for an optimal tree is computationally intractable. Indeed, the Maximum Parsimony problem is extremely difficult to solve since it is equivalent to the NP-complete Steiner problem in the hypercube [7]. This is why heuristics methods constitute the main alternative in order to obtain a near optimal tree with reasonable computation time [12, 15].

In this article, we are interested in finding high quality solutions for the phylogeny reconstruction problem with Maximum Parsimony under Fitch's criterion. We present a memetic algorithm which combines a local searcher using a parametric neighborhood relation and a specific tree crossover based on a topological distance. The crossover operator aims at creating new phylogenetic trees by conserving topological distance properties of parent trees while the local searcher improves the quality of each newly created offspring by a multi-neighborhood descent algorithm.

In order to accelerate the fitness evaluation of potential solutions (phylogenetic trees) which is a critical issue in our context, we introduce a vectorization technique to fully take advantage of some characteristics offered by modern x86 processors. To our knowledge, this technique was never described in the literature and helps decrease the computation time by about 80% on any x86 modern processor.

To assess the performance of the proposed memetic algorithm, we report computational results on a set of 12 benchmark instances from TreeBase, a well-known database on phylogenetic information and compare these results with

those obtained by TNT, which is considered to be the most efficient software for the MP problem.

The rest of the paper is organized as follows. In the next section we formally introduce the MP problem. We then describe the main components of the memetic algorithms (Section 3). Computational results and comparisons are shown in Section 4 where some analysis is also given to study the influence of some basic ingredients on the performance of the algorithm (Section 4.4).

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 [5]).
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 $\sum \phi(S_w), \forall w \in I$.*

Remark: For a given phylogenetic tree, its parsimony score can be efficiently (in polynomial time) calculated by the Fitch's algorithm given in [5]. However, it is primordial to accelerate the scoring process as much as possible within a search algorithm when a great number of potential trees must be evaluated.

Definition 4 (Maximum Parsimony Problem - MPP). *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 A Memetic Algorithm for the MPP

3.1 Outline of the algorithm

The algorithm presented in this paper follows a simple, yet powerful memetic schema. From an initial population of solutions, the algorithm carries out a number of evolution cycles. At each cycle, two parents are selected and a crossover operator is applied to the parents to create a new solution (Section 3.2). Each newly created offspring undergoes some local improvements (Section 3.4) before being inserted into the population. Other issues to be considered concern among others fitness evaluation (Section 3.3), parents selection and insertion condition. Parents selection operates with a tournament selection strategy. Two groups of 20% of the individuals are first constituted and two individuals that represent the best individuals of each group are then selected. A new individual T is inserted into the population if it is not already present in the population. The individual that will be removed from the population can be the oldest or the closest to the new one. The outline of the Hydra algorithm is given on Algorithm 1. We describe below the basic ingredients of this memetic algorithm for the MPP.

3.2 Initial population

The population is composed of phylogenetic trees (individuals) for the given set of taxa. The individuals of the initial population are generated with a greedy algorithm. At each step of the algorithm, an isolated taxon (future leaf) is randomly selected and inserted on a branch of the current tree such that this insertion position minimizes Fitch's score. This corresponds to the 1stRotuGbr heuristics used in [1]. Notice that each initial individual undergoes improvements using the local searcher (Section 3.4).

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

input: A : a set of aligned taxa, N : the population size, M number of LS iterations
output: The most parsimonious tree found

P = *GeneratePopulation(A,N)*
for a given number of generations **do**
  $(T_1, T_2) \leftarrow$ *ChooseParents(P)*
  $T \leftarrow$ *Recombination*(T_1, T_2) // *to generate a new tree*
  $T \leftarrow$ *Local_search_improvement(T,M) // to improve an offspring*
  $P \leftarrow$ *Replace(P,T) // To insert the new offspring*
end for
return the best tree found

3.3 Fitness evaluation

Each individual of the population is evaluated according to the Definition 1 and by Fitch's algorithm. Such an evaluation assigns to the individual a parsimony score. Given that the fitness evaluation is one of the most time consuming elements of the algorithm, we use a specific implementation technique to accelerate this evaluation (see Section 3.6)

3.4 Local search with progressive neighborhood

Our local searcher is based on a descent algorithm using a powerful neighborhood called *Progressive Neighborhood* (PN) introduced in [10]. Similar to the VNS (Variable Neighborhood Search) heuristic, PN modifies the size of the neighborhood during the search. But, contrary to VNS, PN starts from a medium (or large) size neighborhood like SPR or TBR [20] and progressively reduces it to the small size NNI [21] neighborhood using a parametric neighborhood relation that evolves during the search. Experiments presented in [10] showed the efficiency of this progressive neighborhood in comparison with other existing neighborhoods for phylogenetic reconstruction. The basic rational behind PN is 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 only minor refinements are necessary at the end of the local search process where a neighborhood like NNI is sufficient.

In order to make the neighborhood evolve, a topological distance on trees is defined in [10] that enables to build a distance matrix for a set of taxa given a tree topology. This distance is also used to control the size of the neighborhood (i.e. the distance between a pruned edge and its inserted edge is at most equal to a given distance).

Definition 5 (Topological distance). *Let i and j be two taxa of a tree T. The topological distance $\delta_T(i, j)$ between i and j is defined 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.*

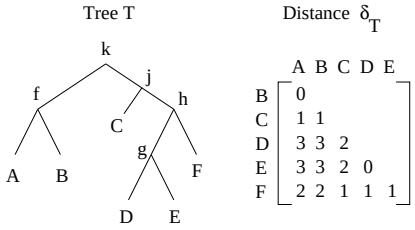


Fig. 1: Example of topological distance δ_T

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 ($f \rightarrow k \rightarrow j \rightarrow h \rightarrow g$), 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 process used in Hydra takes into account a parameter M which corresponds to a maximum number of LS iterations. Using this distance definition, a parameter d is introduced to control 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). As such, changing d leads to neighborhoods of different sizes which are explored with a descent algorithm.

3.5 Distance-preserving crossover operator DiBIPX

For phylogeny reconstruction, the individuals are trees. To create new solutions by recombination, we need a tree crossover operator. Traditionally, crossovers on phylogenetic trees follow the *subtree cutting and regrafting* strategy which consists in removing a subtree T_1^* from a parent T_1 and to reinsert it in the second parent T_2. Duplicated sequences (leaves) are removed from T_2. However, this process suffers from 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 those crossovers perform a *blind* search and it would then be necessary to be able to pass in review millions of trees per second to keep up with the search efficiency.

For the purpose of creating meaningful offspring, we use the Distance-Based Information Preservation tree crossover (DiBIPX) introduced in [9]. DiBIPX is different from conventional tree crossover operators in that it explicitly takes into account the topological distance of parents trees. The key idea behind DiBIPX 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. More precisely, DiBIPX is composed of three steps. First, using the topological distance (Section 3.4), the parents trees are first transformed into two distance matrices M_1 and M_2 (see Figure 1). These matrices are then combined into a new distance matrix M_3 by

a matrix operation $\oplus$. Finally, this new matrix M_3 is used to built the offspring tree using a distance-based clustering method like UPGMA [18]. In this paper, $M_3 = \alpha M_1 \oplus (1-\alpha)M_2$ such that $\forall \delta_1(i,j) \in M_1$ and $\delta_2(i,j) \in M_2$, $\delta_3(i,j) = \alpha \times min\{\delta_1(i,j), \delta_2(i,j)\} + (1-\alpha) \times max\{\delta_1(i,j), \delta_2(i,j)\}$ with $\alpha \in [0,1]$.

3.6 Acceleration of the fitness evaluation

During the resolution of the MPP, the basic operation that is extensively used is Fitch's function (see Definition 1), which is implemented figure 2 (in C-like style). This function takes as input two taxa `t1` and `t2`. The output is the hypothetical taxon `t3` and the number of changes returned by the function. 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.

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

Fig. 2: Parsimony function traduction of Definition 1

Our implementation of this function takes full advantage of some relevant features offered by modern x86 processors. More precisely, the core of modern x86 processors has a SSE (SIMD Streaming Extension) unit which enables to vectorize the code. A SSE register is 128 bits long and can contain 16 bytes. 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 $31 = 1 + 2 + \cdots + 16$. The union can be performed by the binary-OR (`|`) and the intersection by the binary-AND (`&`). The vectorization of Fitch's function gives a 90% improvement on Intel Core 2 Duo processors, while other architectures (pentiumII/III/4, pentium-M, Athlon 64, Sempron) provide 70 to 80 % improvement. This improvement enabled us to divide the overall computation time of our program by a factor of 3 to 4.

4 Computational results and comparisons

In this section, we show computational results on a set of 12 benchmark instances. To assess their quality, they are compared with the results of the very

popular software TNT [11] (Tree analysis using New Technology). (TNT is perhaps today's most powerful software for phylogenetic reconstruction under PM criterion). The code of TNT is not an open source, but is known to be highly optimized to be able to evaluate millions of trees per second. TNT integrates a large number of search strategies such as tree drifting, parsimony ratchet, sectorial search and more others. The reference software TNT was used with its default parameters specified in the documentation.

4.1 Benchmarks

A set of 12 medium to large instances (more than 100 taxa) from the TreeBase site (www.treebase.org) are used. TreeBase is a relational database of phylogenetic information which stores phylogenetic trees and the data matrices used to generate them from published research papers. For these instances, no optimum parsimony score is known.

4.2 Parameters tuning

In order to carry out the comparisons as fair as possible, we use for Hydra a set of standard (not fine-tuned) parameters values as follows for all the tested instances. With these parameters, each run of the algorithm undergoes $(40+50)\times 200,000 = 18\times 10^6$ iterations (fitness evaluations with Fitch's algorithm). As we shall see later on, this number is much less than the one reached by the reference software TNT with its default parameters. According to the problem instance, the running time of one execution of Hydra goes from one to ten minutes on a Core 2 at 2.5MHz computer. Nevertheless, TNT consumes less computation time due to its extremely optimized code.

Parameters	Values
population size	40
number of crossovers (generations)	50
local search iterations per crossover	200 000
neighborhood	progressive
α for DiBIP crossover	0.70

Table 1: Parameters of the Hydra algorithm.

4.3 Results and comparisons

On table 2 we report the results obtained with Hydra. For each instance, we report the following information for 20 executions: best parsimony score, maximum parsimony score, average score and standard deviation. For TNT, the software

			Hydra				TNT	
problem	#taxa	length	min	max	avg	std	# iter	score
m0808	178	3453	**23,777**	23,794	23,782.22	4.62	40	23,782
m0972	155	355	**1,528**	1,529	1,528.44	0.50	14	1,532
m1038	297	2021	**12,370**	12,375	12,371.22	2.19	99	12,376
m1902	209	977	**6,124**	6,129	6,126.22	1.52	44	6,131
m1987	134	618	**2,073**	2,075	2,073.67	0.92	30	**2,073**
m2055	299	2064	**2,597**	2,604	2,599.00	2.37	47	2,598
m2123	198	1426	**1,925**	1,925	1,925.00	0.00	46	**1,925**
m2725	210	8245	157,354	157,772	157,570,56	95.29	63	**157,093**
m2780	119	2020	**11,488**	11,494	11,490.78	1.90	**11**	11,490
m3275	117	5098	**21,935**	21,936	21,935.11	0.30	**9**	**21,935**
m3452	116	1157	**3,602**	3,603	3,602.33	0.49	23	**3,602**
m3453	137	995	**3,904**	3,908	3,904.78	1.42	24	**3,904**

Table 2: Comparison between Hydra and TNT on TreeBase instances.

gives only two information: the best parsimony score and the total number of iterations (which corresponds to the number of evaluated candidate trees).

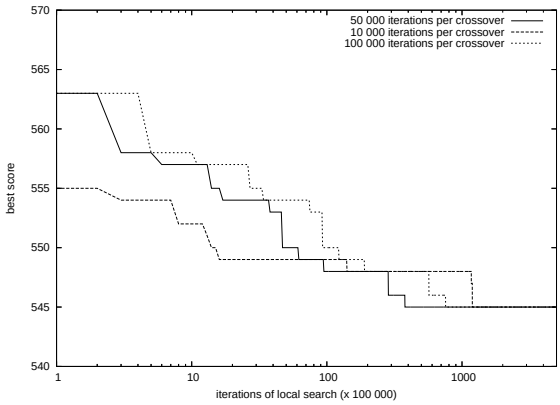
The results show that except for one instance, Hydra obtains results of equal or better quality than TNT. Hydra reaches these results with much smaller iterations (much fewer evaluations of candidate trees) except for problems *m0972*, *m2780* and *m3275* for which the number of iterations of Hydra is greater than TNT. Nevertheless the experiments we have carried out show that by fixing M to 50,000 or 100,000 (i.e. 4.5 or 9 millions iterations), we could obtain the same scores.

Note that the computation time of TNT is less important than Hydra because TNT implements highly specific optimization techniques to avoid a complete evaluation of the overall tree when looking for a neighbor of lower cost. Such technique is not yet implemented in Hydra.

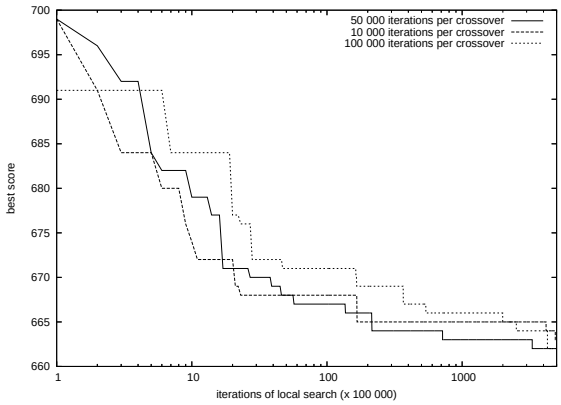
4.4 Analysis and discussion

In this section, we investigate and analyze the influence of some elements of the Hydra algorithm on its performance.

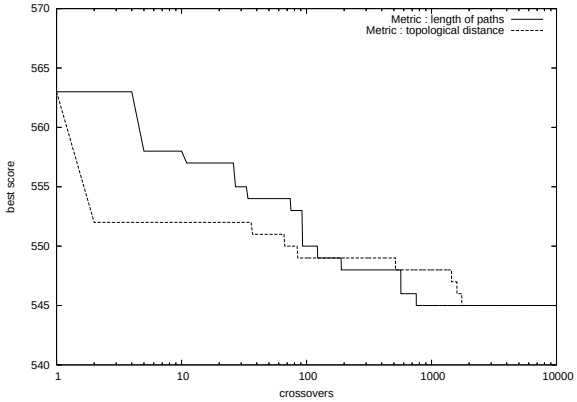
First, we verify how the length of the LS improvement after each crossover impacts on the convergence of the algorithm. To avoid a possible bias due to the structure of problem instance, we chose to use two *random* instances (tst01 and tst20) from [16]. We run the Hydra algorithm with three different values of local search iterations per crossover: 10 000, 50 000 and 100 000. Figure 3 (a-b) shows on a logarithmic scale, the result of the evolution of the best score averaged over 5 independent runs, each run being limited to 10^8 iterations. One observes that if the computational resources are limited, the algorithm converges faster with short local search than with long local search iterations. The influence of the local search iterations seems to decrease when the search progresses.



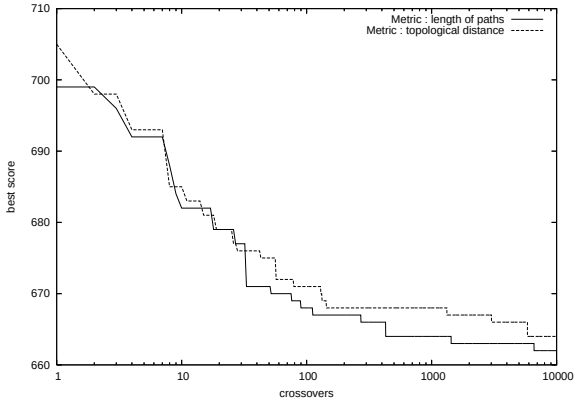
(a) Influence of number of LS iterations - tst01



(b) Influence of number of LS iterations - tst20



(c) Influence of distance metric - tst01



(d) Influence of distance metric - tst20

Fig. 3: Study of influence of three parameters

Second, we check the performance of DiBIPX if the topological distance defined in Section 3.4 is replaced by another distance metric. For this purpose, we use another distance which is the length of the path between two nodes of the tree taking into account the parsimony distance between each pair of neighboring nodes. Figure 3 (c-d) shows the result of the evolution of the best score under the same condition as above. One observes there is no clear dominance of a distance metric over the other one, but the convergence with the topological distance is sometimes faster for a short number of LS iterations.

5 Conclusion

In this paper, we have presented the Hydra memetic algorithm for the phylogenetic tree inference problem with the Maximum Parsimony criterion. The algorithm uses a local search procedure which is based on a parametric progressive neighborhood, a specific distance-based topological tree crossover, as well as a specific implementation technique in order to accelerate the fitness evaluation.

Experimentations on a number of real benchmark instances from TreeBase show that Hydra competes very well when compared to TNT. Indeed, Hydra is able to find phylogenetic trees of better Parsimony score with much fewer evaluations of candidate trees.

However, Hydra needs more computation time than TNT. This last point constitutes an important issue to be addressed in the future. On the other hand, the techniques presented in this paper may be integrated into TNT to increase its search power.

6 Availability:

Hydra is distributed with C++ source code, benchmarks and documentation and is freely available from the website of the authors. It runs under all Unix/Linux platforms. There is also a binary for Windows platforms (http://www.info.univ-angers.fr/pub/richer/rec.php).

7 Acknowledgements

This work was partially suported by the French Ouest Genopole® and by the BIL (Bio-Informatique Ligérienne) project (http://www.info.univ-angers.fr/bil).

References

1. A.A. Andreatta and C.C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics*, 8:429–447, 2002.
2. L.L. Cavalli-Sforza and A.W.F. Edwards. Phylogenetic analysis: models and estimation procedures. *Evolution*, 32:550–570, 1967.
3. A.W.F Edwards and L.L. Cavalli-Sforza. The reconstruction of evolution. *Annals of Human Genetics*, 27:105–106, 1963.
4. J. Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17:368–376, 1981.
5. W. Fitch. Towards defining course of evolution: minimum change for a specified tree topology. *Systematic Zoology*, 20:406–416, 1971.
6. W.M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155(3760):279–284, 1967.
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. On the optimization principle in phylogenetic analysis and the minimum evolution criterion. *Biology and Evolution*, 17:401–405, 2000.

9. 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.
10. 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*, Vol 5 (1) January-March 2008.
11. P. A. Goloboff, J. S. Farris, and K. Nixon. Tnt : Tree analysis using new technology. http://www.cladistics.com/aboutTNT.html, 2003.
12. P.A. Goloboff. Character optimisation and calculation of tree lengths. *Cladistics*, 9:433–436, 1993.
13. D.M. Hillis, C. Moritz, and B.K. Mable. *Molecular Systematics*. Sinauer Associates, Inc., 1996.
14. J.H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, 1975.
15. K.C. Nixon. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15:407–414, 1999.
16. 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.
17. N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.
18. R.R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958.
19. R.R. Sokal and P.H.A. Sneath. *Principles of Numerical Taxonomy*. W.H. Freeman, San Francisco, 1963.
20. D. L. Swofford, G. J. Olsen. in D.M. Hillis and C. Moritz (Ed.) Phylogeny Reconstruction. *Molecular Systematics*, chapter 11:411-501, 1990.
21. M. S. Waterman and T. F. Smith. On the similarity of dendograms. *Journal of Theoretical Biology* 73:789-800, 1978.