A Bottom-Up implementation of Path-Relinking for Phylogenetic Reconstruction applied to Maximum Parsimony

Karla E. Vázquez-Ortiz
LERIA
2 Boulevard Lavoisier
49045 Angers Cedex 01, France
Email: vazquez@info.univ-angers.fr

David Lesaint
LERIA
2 Boulevard Lavoisier
49045 Angers Cedex 01, France
Email: lesaint@info.univ-angers.fr

Jean-Michel Richer
LERIA
2 Boulevard Lavoisier
49045 Angers Cedex 01, France
Email: richer@info.univ-angers.fr

Eduardo Rodríguez-Tello
CINVESTAV-Tamaulipas, Information Technology Laboratory
Km. 5.5 Carretera Victoria-Soto La Marina
87130 Victoria Tamps., Mexico.
Email: ertello@tamps.cinvestav.mx

***Abstract*—In this article we describe a bottom-up implementation of Path-Relinking for Phylogenetic Trees in the context of the resolution of the Maximum Parsimony problem with Fitch optimality criterion. This bottom-up implementation is compared to two versions of an existing top-down implementation. We show that our implementation is more efficient, more interesting to compare trees and to give an estimation of the distance between two trees in terms of the number of transformations.**

***Keywords*—Phylogenetic Reconstruction, Path-Relinking**

I. INTRODUCTION

Path-Relinking (PR) is a metaheuristic with an intensification strategy used to explore elite solutions. It was defined by Fred Glover [1] and it is closely related to Tabu Search. PR has been applied to a variety of contexts where it has proved to be very effective in solving difficult problems. Given two solutions called source and guiding, PR consists in transforming the source solution into the guiding solution by applying a series of modifications. After each modification an exploration phase is performed on a copy of the source solution under modification. The aim of PR is to generate a path from the source to the guiding solution in order to possibly find a better solution. PR has been used in different domains for the resolution of Combinatorial Optimization problems [2], [3], [4], [5].

In [6] the authors describe an implementation of PR tailored to trees for the resolution of the Maximum Parsimony problem. This implementation can be called *top-down* implementation as it starts from the root of the tree and recursively explore each left and right subtree. For each iteration they compare the taxa in the left and right subtrees of the source and guiding solutions. All taxa of the source left (resp. right) subtree that are in the right (resp. left) subtree of the guiding solution are moved to the right (resp. left) subtree of the source solution. The major drawback of this implementation is that it requires a lot of modifications and moves of the taxa from left to right (resp. right to left) subtrees. When a taxon is degraphed it is then regraphed on a branch of the sibling subtree that minimizes the parsimony score of the tree. We can also put the degraphed taxon at the top of the sibling subtree to avoid the search of a minimum tree. If we remove the exploration phase from the PR algorithm we then have an algorithm that transforms one tree into another which can be used to compare trees and define a measure of distance between them. This measure of distance seems to be interesting because it is closer to the topology of the trees to compare.

The remainder of the article is organized as follows: in the next section we explain the basics of Maximum Parsimony that will serve for our experimentations. In section 3, we describe the method that we have designed to perform Path-Relinking between two trees based on the difference of subtrees. The last section is dedicated to computational results and analysis of the implementations.

II. MAXIMUM PARSIMONY

One of the main problems in Comparative Biology consists in establishing ancestral relationships between a group of n species or homologous genes in populations of different species, designated as taxa (or operational taxonomic units). These ancestral relationships are usually represented by a binary rooted tree, which is called a phylogenetic tree or, in short, a phylogeny [7]. In the past phylogenetic trees were inferred by using morphological characteristics like color, size, number of legs, etc. Nowadays, they are improved using the information from biologic macromolecules like DNA (deoxyribonucleic acid), RNA (ribonucleic acid) and proteins. The problem of reconstructing molecular phylogenetic trees has become an important field of study in Bioinformatics and has many practical applications in population genetics, whole genome analysis, and the search for genetic predictors of disease [8], [9].

There exist many different methods reported in the literature to solve the problem of phylogenetic reconstruction. These methods can be classified into three main approaches: *Distance methods* [10], [11], *Probabilistic methods* [12], [13] and *Cladistic methods* [14], [15]. In this paper we will use a cladistic method based on Maximum Parsimony (MP) with Fitch optimality criterion but it can be applied to any of the other approaches cited above and more generally to any method that deals with trees. With MP we are looking for the tree that minimizes the amount of evolution (in terms of number of mutations). This approach is based on the assumption that each character (characteristic feature) evolves independently. For a thorough introduction to the MP problem we refer the reader to [16].

A. Problem statement

Let $\mathscr{S}$ be a set $\{S_1, S_2, \ldots, S_n\}$ composed of n sequences of length k over a predefined alphabet $\mathscr{A}$. A *binary rooted phylogenetic tree* $t = (V, E)$ is used to represent their ancestral relationships. It consists of a set of nodes $V = \{v_1, \ldots, v_r\}$ and a set of edges $E \subseteq V \times V = \{\{u, v\} | \, u, v \in V\}$. The set of nodes V ($|V| = (2n-1)$) is partitioned into two subsets: I that contains $n-1$ *internal nodes* (or hypothetical ancestors) each having 2 descendants; and L which is composed of the n *taxa* that are leaves of the tree, i.e. nodes with no descendant. For each subtree $v_z = (v_x, v_y)$ such that v_z is the root and $\{v_z, v_x\}, \{v_z, v_y\} \in E$ we consider the sequences related to each node and call them respectively x, y and z. The hypothetical parsimony sequence $z = \{z_1, \cdots, z_k\}$ is computed from $x = \{x_1, \cdots, x_k\}$ and $y = \{y_1, \cdots, y_k\}$ with the following expression:

$$\forall i, 1 \leq i \leq k, z_i = \left\{ \begin{array}{ll} x_i \cup y_i, & \text{if } x_i \cap y_i = \emptyset \\ x_i \cap y_i, & \text{otherwise} \end{array} \right. \quad (1)$$

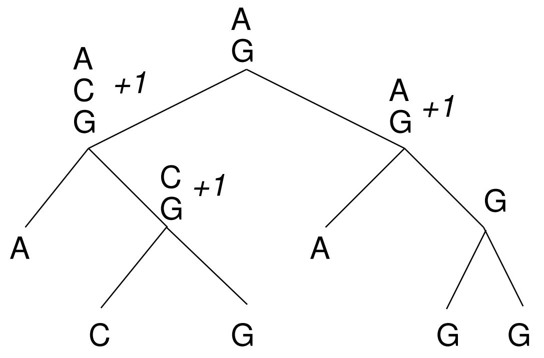
Then, the parsimony cost of the sequence z under Fitch optimality criterion [17], [18] is defined as follows:

$$\phi(z) = \sum_{i=1}^{k} C_i \qquad \text{where} \qquad C_i = \left\{ \begin{array}{ll} 1, & \text{if } x_i \cap y_i = \emptyset \\ 0, & \text{otherwise} \end{array} \right. \quad (2)$$

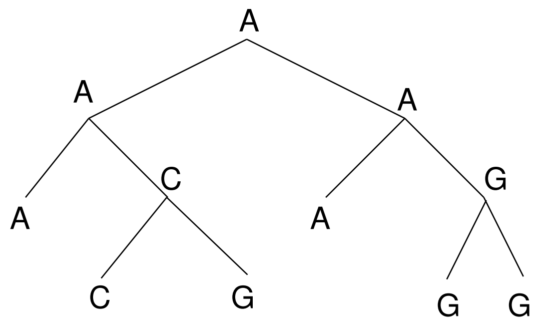
and the parsimony cost of the tree t is obtained as follows:

$$\phi(t) = \sum_{z \in I} \phi(z) \quad (3)$$

The MP problem is then a minimization problem which consists in finding a tree topology t^* such that $\phi(t^*)$ is minimum, i.e., $\phi(t^*) = \min\{\phi(t) : t \in \mathscr{T}\}$, where $\mathscr{T}$ is the set composed of all the possible tree topologies also known as the search space of the problem. In order to compute the overall cost (or score) $\phi(t)$, which is the objective function of the problem, Fitch's algorithm [17] gradually moves back from the leaves towards the root and computes hypothetical ancestral taxa z for each internal node in I. This is often referred to as the *first pass* of the algorithm (see Fig. 1) whose complexity is $O(n \times k)$. A *second-pass* can eventually be used to assign one nucleotide to a site even if many possibilities exist, in order to obtain a hypothetical tree.



(a) first pass from leaves to root



(b) second pass from root to leaves

Fig. 1: First-pass and assignment of states after second-pass for a tree of score 3 under Fitch's optimality criterion with $n = 6$ taxa of length $k = 1$

B. Resolution

It has been demonstrated that the MP problem is NP-complete [19], [20], since it is equivalent to the Combinatorial Optimization problem known as the Steiner tree problem on hypercubes. This essentially means that no algorithm that solves all instances quickly is likely to be found. Indeed, for a set of n taxa, the number of rooted tree topologies is given by the following expression [21]:

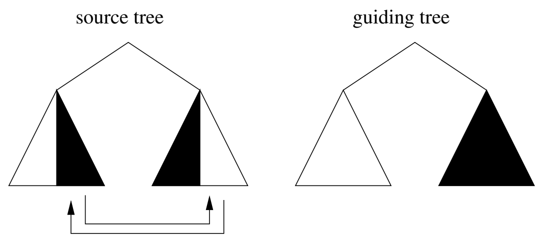
$$|\mathscr{T}| = (2n-3)!/2^{n-2}(n-2)!$$

The MP problem has been exactly solved for very small instances ($n \leq 10$) using a *branch & bound* algorithm (B&B) originally proposed by Hendy and Penny [22]. One of the most recent applications of B&B to the MP problem is XMP, a program for finding exact MP trees which uses optimized vectorized inner loops [23] on highly parallel distributed-memory computers. For their experiments the authors used real and synthetic instances from [24] and other real datasets, these instances have between 12 and 36 taxa which are very small instances. The best approaches to solve MP and reach interesting solutions are based on metaheuristics. Many software packages have been developed (PAUP [25], POY [26]) and are based mainly on *local search*. Some methods use *genetic or memetic algorithms* (Hydra [27]) or *simulated annealing* (SAMPARS [28]) or a combination of meta-heuristics and other techniques (TNT [29]). A very simple method is GRASP (Greedy Randomized Adaptative Search Procedure), a multi-start metaheuristic for which a set of initial solutions is generated and are then improved by a local search. Ribeiro and Vianna [30] have applied GRASP with VNS (Variable Neighborhood Search)[31], [32]. The principle of the VNS metaheuristic is to successively use different neighborhoods during a descent. It starts with a neighborhood of small size and once the search is stuck in a local optimum, it uses a neighborhood of larger size in order to allow important modifications of the current solution and escape from the local optimum. An interesting extension of GRASP, which can be considered as a genetic or memetic algorithm is to use PR to try to discover new elite solutions from the set of final solutions obtained by GRASP [6]. PR can also be used with a genetic or memetic algorithm.

III. The method

For the MP problem an implementation of PR was given by [6]. This **top-down recursive** implementation (see Figure 2) starts from the root of the tree and compares the left and right subtrees of the source and guiding solution. All taxa of the left subtree of the source that are present in the right subtree of the guiding solution are moved to the right subtree of the source solution and conversely.

Fig. 2: Top-Down implementation



The major drawback of this implementation (see the results section) is that it requires a lot of modifications and moves of the taxa from left (resp. right) to right (resp. left) subtrees.

In contrast, we have implemented a **bottom-up iterative** solution which compares the subtrees present in the source and guiding solutions (see Figure 3). For this the subtrees of each solution are ordered by their number of leaves and we start to compare subtrees of size 2, then subtrees of size 3, and so on (see Algorithm 1).

When a subtree of the guiding solution $t = (X, Y)$ is not found in the source solution then we have to transform the subtree in the source solution into the one in the guiding solution. Here X is the left subtree of t and Y its right subtree.

Consider the example of Figure 3 (a) where we can see the source and guiding trees. A preliminary modification of the trees (line 1 and 2 of the algorithm) consists in reordering the trees by the lexicographic order of the leaves in order to be able to efficiently compare the subtrees. In fact the subtrees (A, B) and (B, A) are equal and we do not want to have to check both cases, so we will only allow subtrees of the form (A, B).

On each node, one of the sequences on the left subtree must be inferior to all the sequences on the right subtree. For example, the node (B, A) will be changed by swapping the leaves in order to obtain (A, B). The subtree $(C,(A, B)))$ will be reordered as $((A, B), C)$ because on the right node, A and B are inferior to C. The subtree $((C, E),(A, F))$ will be reordered as $((A, F),(C, E))$ because A that initially appears on the right subtree is inferior to C and E.

Algorithm 1: Path-Relinking with bottom-up iterative implementation

```
input: $s$: source tree, $g$ : guiding tree
output: number of transformations
1  $reorder(s)$ ;
2  $reorder(g)$ ;
3  $transformations \leftarrow 0$;
4  $\Omega_g \leftarrow$ ordered set of subtrees of guiding tree $g$;
5  $change \leftarrow true$;
6  while $change$ do
7      $\Omega_s \leftarrow$ ordered set of subtrees of source tree $s$;
8      $change \leftarrow false$ ;
9      if $\exists \quad t = (X, Y) \in \Omega_g - \Omega_s$ then
10         $change \leftarrow true$ ;
11         degraph $Y$ and regraph on $X$ in $s$;
12         $transformations \leftarrow transformations + 1$;
13 return $transformations$
```

TABLE I: Sets of subtrees in the source and guiding trees ordered by number of leaves

#	Source	Guiding
1	A, B, C, D, E, F	A, B, C, D, E, F
2	(A, F), (C, E), (B, D)	(A, B), (E, F)
3		$((A, B), C)$, $(D,(E, F))$
4	$((A, F),(C, E))$	
5		
6	$(((A, F),(C, E)),(B, D))$	$(((A, B), C),(D,(E, F)))$

The subtrees present in the source and guiding solutions are represented in Table I in Newick notation. After reordering the source and guiding trees we can enter the main loop of the algorithm (lines 6 to 12).

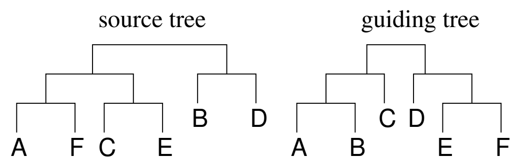
The subtree (A, B) present in the guiding tree is not present in the source tree so we degraph B from the source tree, and regraph it on A (see Fig. 3b).

On Figure 3b the subtree (E, F) present in the guiding tree is not present in the source tree so we degraph F from the source tree and regraph it on E.

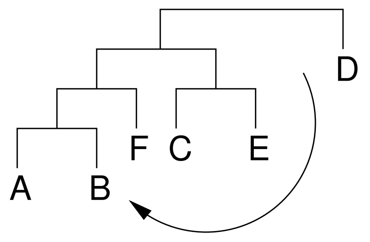
On Figure 3c the subtree $((A, B), C)$ present in the guiding tree is not present in the source tree, we have to degraph C from the source tree and regraph it on (A, B).

The subtree $(D,(E, F))$ present in the guiding tree is not present in the source tree: degraph (E, F) from the source tree and regraph it on D (see Fig. 3e).

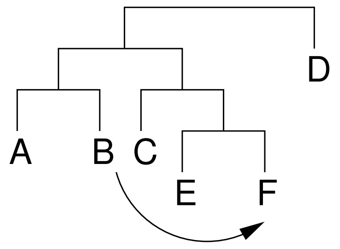
Finally the source and guiding trees are equal so we can stop the algorithm.



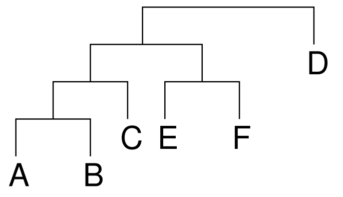
(a) source and guiding trees



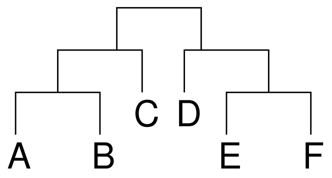
(b) first modification resulting in (A, B) on source



(c) modification resulting in (E, F)



(d) modification resulting in $((A, B), C)$



(e) modification resulting in $(D, (E, F))$

Fig. 3: Example of Bottom-Up Path-Relinking with source and guiding trees

A. *Complexity*

The complexity of the algorithm can be computed as follows: given n, the number of taxa of the problem, the reordering of the guiding tree needs $2n - 1$ comparisons and the computation of Ω_g can be done in $2n - 1$ operations. The main loop will be executed a certain number of times, let's say p times, and we will need to compute Ω_s, find a missing subtree (the maximum will be n comparisons) and perform a degraph and regraph (1 transformation). This adds up to:

$$p \times (2n - 1 + n + 1) + 3 \times (2n - 1) \simeq 3n \times (p + 2)$$

In the worst case $p = n$, so the worst case complexity is $O(n^2)$ for the *bottom-up* implementation.

For the *top-down* implementation, the computation of the complexity is more difficult to establish as the process is recursive and the size of the subtrees changes. At each step we must compute the sets of leaves on the left and right subtrees of the source and guiding solutions. Then move the leaves in the source tree from left to right or from right to left. The number of transformations (see the tables in the results section) can give us some insight of the complexity of this implementation.

IV. Results

A. *Benchmark*

For simplicity's sake we have decided to report results for a problem called zilla by some authors and that was originally obtained from the chloroplast gene rbcL [33]. Other experimentations carried out on some other problems of TreeBase (treebase.org) should likely exhibit the same behavior.

The problem zilla is interesting because it is large (500 taxa of 759 DNA residues) and the best parsimony score of 16,218 which is challenging to achieve was first found by TNT. In Table II we report the percentage of common subtrees between two trees that are compared. The number of common subtrees can be thought as a measure of similarity between trees.

For example two trees of score 16,218 (namely 16,218 and 16,218^b) have 74.75% subtrees in common which means that they are topologically close. On the contrary the tree of score 21,727 has 0% subtrees in common with the tree of score 16,218. This means they are very far from each other and have only the leaves in common.

Three different implementations were compared:

- *bottom-up*: iterative implementation explained in this article, based on subtree difference,
- *top-down without minimization*: recursive implementation from the article of [6] without optimization on regraph, i.e. when a leaf is regraphed it is placed at the root of the subtree where it should appear,
- *top-down with minimization*: recursive implementation from the article of [6] with optimization on regraph, i.e. when a leaf is regraphed all possible branches are tested and we keep the first one that minimizes the score of the tree.

B. *Experiments*

The experimentations were performed on an Intel Core i5 4570 and the program was coded in Java 1.7, it is part of a software called Arbalet (http://www.info.univ-angers.fr/pub/richer/ur.php?arbalet). On tables III, IV, V we report for

TABLE II: Percentage of common subtrees for Parsimony trees of zilla used for Path-Relinking

source / guiding	% common subtrees
16218^b / 16218	74.75%
16219 / 16218	73.15%
16250 / 16218	61.12%
16401 / 16218	45.69%
16611 / 16218	36.27%
21727 / 16218	0.00%
16250 / 16219	61.72%
16401 / 16219	48.70%
16401 / 16250	45.49%
16611 / 16250	44.89%
21727 / 16250	0.20%
16611 / 16401	36.67%
21727 / 16611	0.20%

each implementation the number of transformations (degraph + regraph), the execution time in seconds, the number of times the source tree had a score inferior or equal to the guiding tree (#Equal) and the number of times the source tree had a score strictly inferior to the guiding tree (#Less) during the generation of the path.

Note that the source tree has a higher score than the guiding tree. This is not necessary for the *bottom-up* method for which we can invert the trees. However this is required by the *top-down with minimization* implementation.

TABLE III: Results of Path-Relinking for Bottom-Up implementation (Times in seconds)

	Bottom-Up			
source / guiding	Trans.	Time	#Equal	#Less
16218^b / 16218	24	0.10	13	0
16219 / 16218	32	0.14	2	0
16250 / 16218	97	0.40	3	0
16401 / 16218	151	0.60	2	0
16611 / 16218	186	0.75	2	0
21727 / 16218	446	1.68	2	0
16250 / 16219	92	0.37	2	0
16401 / 16219	152	0.61	2	0
16401 / 16250	162	0.63	1	0
16611 / 16250	144	0.56	1	0
21727 / 16250	449	1.71	1	0
16611 / 16401	202	0.79	3	0
21727 / 16611	455	1.84	2	0

TABLE IV: Results of Path-Relinking for Top-Down without minimization on regraph (Times in seconds)

	Top-Down No Minimization			
source / guiding	Trans.	Time	#Equal	#Less
16218^b / 16218	118	0.33	16	0
16219 / 16218	462	1.36	1	0
16250 / 16218	1770	4.79	1	0
16401 / 16218	1891	5.16	1	0
16611 / 16218	1903	5.19	1	0
21727 / 16218	1881	5.17	1	0
16250 / 16219	1722	4.76	1	0
16401 / 16219	1867	6.17	1	0
16401 / 16250	1707	4.71	1	0
16611 / 16250	1746	4.79	1	0
21727 / 16250	1712	4.77	1	0
16611 / 16401	1602	4.42	1	0
21727 / 16611	1842	5.16	1	0

TABLE V: Results of Path-Relinking for Top-Down with minimization on regraph (Times in seconds)

	Top-Down With Minimization			
source / guiding	Trans.	Time	#Equal	#Less
16218^b / 16218	74	0.85	21	0
16219 / 16218	343	4.08	3	0
16250 / 16218	1519	25.32	2	0
16401 / 16218	1474	25.36	2	0
16611 / 16218	1225	19.57	2	0
21727 / 16218	1241	18.70	2	0
16250 / 16219	1418	24.70	1	0
16401 / 16219	1390	23.74	1	0
16401 / 16250	1203	19.74	1	0
16611 / 16250	1233	35.78	3	0
21727 / 16250	1196	19.65	1	0
16611 / 16401	1216	47.75	2	1
21727 / 16611	1528	22.40	593	586

On Table III, with the *bottom-up* implementation the path between the two trees of score 16,218 (called 16,218 and 16,218^b) was built with 24 transformations in 0.1 seconds. As mentioned before those trees are topologically very close. During the transformation process the source tree, when modified, had a best score (of 16,218) 13 times among the 24 transformations.

With the *top-down with minimization* algorithm (see results of Table V) the construction of the path of the tree of score 21,727 into the tree of score 16,611 has needed 1528 transformations and took 22.4 seconds. It also lead to the generation of 586 trees of score under 16,611. This means that the *top-down with minimization* algorithm can sometimes help find a tree of lower score than the guiding tree.

C. Discussion

1) Execution time: although the *bottom-up* implementation requires to recompute the set of subtrees of the source tree for each iteration it is the fastest method. The *top-down with minimization* implementation takes much more time because of the optimization phase on regraph. However for the *top-down* implementations the number of transformations in terms of nodes to degraph and regraph is very important compared to the *bottom-up* version. For example for the search of a path from tree 21,727 to 16,218, the number of recursive calls is equal to 499. The number of leaves moved for the first 10 steps of the recursion are: 194, 143, 21, 3, 15, 34, 40, 4, 43, 3, ... for a total of 1881 transformations.

2) Comparison of trees: it seems that the *bottom-up* implementation is more suitable to compare trees than the *top-down* of [6] because it reports a number of transformations proportional to the topological modifications of the tree. For example, on Table III the number of transformations from the tree of score 16,219 to 16,218 is 32, while the number of transformations from the tree of score 21,727 to 16,218 is 455. For the *top-down implementation without minimization* the transformation from the tree of score 16,401 to 16,218 is nearly equal to the number of transformations of the tree of score 21,727 to 16,218.

There exists different metrics and methods to compare trees. One of the most famous is the Robinson-Foulds metric [34] initially designed for unrooted tree but which has a variant for rooted trees based on clusters [35]. RF is is equal to the number of different splits in compared trees. A split $A|B$ of a set L is an unordered pair (ie, $A|B = B|A$) of its subsets, such that $L = A \cup B$ and $A \cap B = \emptyset$. Our method can to some extent be brought closer to the Matching Cluster metric for rooted trees but provides different results. [35].

3) Ability to find better solutions: finally, we can remark that with no exploration phase PR generally does not allow us to find a tree with a score lower than the guiding tree (see columns #Less and #Equal), except for the *top-down with minimization* version but only for trees that are far from the best known solution. It seems to become harder to find improving solutions as long as we come close to the best know solution. In the case of the search of a path from tree 21,727 to 16,611 we have found that the tree with the lowest score was 16,532.

Nevertheless it is possible to add a minimization phase to the *bottom-up* algorithm: the new subtree $t = (X, Y)$ obtained from line 11 of Algorithm 1 can be degraphed and regraphed somewhere on the tree in order to minimize the parsimony score. However t must not be regraphed on some previously modified subtree in order to avoid any cycle of transformation that would cause a modification to be undone and that would be performed again at the next iteration. We have tested this method but it lead to no improvement and we could not find a score inferior to the score of the guiding tree.

V. Conclusion

In this paper we have described an implementation of Path-Relinking in the context of Phylogenetic Reconstruction with Maximum Parsimony. Confronted to other existing implementations our method does not allow to find trees with a better score which is the aim of Path-Relinking but represents an interesting tool to compare the topologies of the source and guiding trees. The *bottom-up iterative* implementation what we have described is faster than the *top-down recursive* implementations and can serve as a measure of distance between trees and could be applied to any other context.

References

[1] F. Glover, M. Laguna, and R. Mart, “Fundamentals of scatter search and path relinking,” *Control and Cybernetics*, vol. 39, pp. 653–684, 2000.

[2] Y. Wang, Z. L, F. Glover, and J.-K. Hao, “Path relinking for unconstrained binary quadratic programming,” *European Journal of Operational Research*, vol. 223, no. 3, pp. 595–604, 2012. [Online]. Available: http://EconPapers.repec.org/RePEc:eee:ejores:v:223:y:2012:i:3:p:595-604

[3] K. Seridi, L. Jourdan, and E.-G. Talbi, “Multi-objective path relinking for biclustering: application to microarray data,” in *EMO'2013 Evolutionary Multi-objective Optimization*, Sheffield, Royaume-Uni, March 2013, pp. 200–214. [Online]. Available: http://hal.inria.fr/hal-00837571

[4] H. Z. Yongquan Zhou, Jian Xie, “A hybrid bat algorithm with path relinking for capacitated vehicle routing problem,” 2013. [Online]. Available: http://www.hindawi.com/journals/mpe/2013/392789/

[5] M. Resende and C. Ribeiro, “Grasp: Greedy randomized adaptive search procedures,” in *Search Methodologies*, E. K. Burke and G. Kendall, Eds. Springer US, 2014, pp. 287–312. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-6940-7_11

[6] C. C. Ribeiro and D. S. Vianna, “A hybrid genetic algorithm for the phylogeny problem using path-relinking as a progressive crossover strategy,” *International Transactions in Operational Research*, vol. 16, no. 5, pp. 641–657, 2009.

[7] W. Hennig, *Phylogenetic systematics*, ser. Phylogeny. Urbana: University of Illinois Press, 1966.

[8] D. M. Hillis, C. Moritz, and B. K. Mable, *Molecular systematics*, 2nd ed. Sunderland, MA: Sinauer Associates Inc., 1996.

[9] S. Sridhar, F. Lam, G. E. Blelloch, R. Ravi, and R. Schwartz, “Direct maximum parsimony phylogeny reconstruction from genotype data,” *BMC Bioinformatics*, vol. 8, no. 472, 2007.

[10] W. M. Fitch and E. Margoliash, “A method for estimating the number of invariant amino acid coding positions in a gene using cytochrome c as a model case,” *Biochemical Genetics*, vol. 1, no. 1, pp. 65–71, 1967.

[11] N. Saitou and M. Nei, “The neighbor-joining method: a new method for reconstructing phylogenetic trees,” *Molecular Biology and Evolution*, vol. 4, no. 4, pp. 406–425, 1987.

[12] J. Felsenstein, “Evolutionary trees from DNA sequences: a maximum likelihood approach,” *Journal of Molecular Evolution*, vol. 17, no. 6, pp. 368–376, 1981.

[13] A. Skourikhine, “Phylogenetic tree reconstruction using self-adaptive genetic algorithm,” in *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering*, Arlington, VA , USA, 2000, pp. 129–134.

[14] A. W. F. Edwards and L. L. Cavalli-Sforza, “The reconstruction of evolution,” *Heredity*, vol. 18, p. 553, 1963.

[15] L. L. Cavalli-Sforza and A. W. F. Edwards, “Phylogenetic analysis. models and estimation procedures,” *The American Journal of Human Genetics*, vol. 19, no. 3 Pt 1, pp. 233–257, 1967.

[16] J. Felsenstein, *Inferring phylogenies*. Sinauer Associates, 2003.

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

[18] J. A. Hartigan, “Minimum mutation fits to a given tree,” *Biometrics*, vol. 29, pp. 53–65, 1973.

[19] D. Gusfield, *Algorithms on strings, trees, and sequences: Computer science and computational biology*, 1st ed. Cambridge University Press, 1997.

[20] L. R. Foulds and R. L. Graham, "The steiner problem in phylogeny is np-complete," *Advances in Applied Mathematics*, vol. 3, no. 1, pp. 43–49, 1982.

[21] J. Xiong, *Essential Bioinformatics*, 1st ed. Cambridge University Press, 2006.

[22] M. D. Hendy and D. Penny, "Branch and bound algorithms to determine minimal evolutionary trees," *Mathematical Biosciences*, vol. 59, no. 2, pp. 277–290, 1982.

[23] W. T. J. White and B. R. Holland, "Faster exact maximum parsimony search with xmp," *Bioinformatics*, vol. 27, no. 10, pp. 1359–1367, 2011.

[24] D. A. Bader, V. P. Chandu, and M. Yan, "ExactMP: An efficient parallel exact solver for phylogenetic tree reconstruction using maximum parsimony," in *Parallel Processing, 2006. ICPP 2006. International Conference on*. IEEE, 2006, pp. 65–73.

[25] D. L. Swofford, "PAUP*: phylogenetic analysis using parsimony, version 4.0b10," 2011.

[26] A. Varn, L. S. Vinh, and W. C. Wheeler, "POY version 4: phylogenetic analysis using dynamic homologies," *Cladistics*, vol. 26, no. 1, pp. 72–85, 2010. [Online]. Available: http://dx.doi.org/10.1111/j.1096-0031.2009.00282.x

[27] J. M. Richer, A. Goëffon, and J. K. Hao, "A memetic algorithm for phylogenetic reconstruction with maximum parsimony," *Lecture Notes in Computer Science*, vol. 5483, pp. 164–175, 2009.

[28] J.-M. Richer, E. Rodriguez-Tello, and K. Vazquez-Ortiz, "Maximum parsimony phylogenetic inference using simulated annealing," in *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation II*, ser. Advances in Intelligent Systems and Computing, O. Schtze, C. A. Coello Coello, A.-A. Tantar, E. Tantar, P. Bouvry, P. Del Moral, and P. Legrand, Eds. Springer Berlin Heidelberg, 2013, vol. 175, pp. 189–203. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31519-0_12

[29] P. A. Goloboff, J. S. Farris, and K. C. Nixon, "TNT, a free program for phylogenetic analysis," *Cladistics*, vol. 24, no. 5, pp. 774–786, 2008.

[30] 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*, vol. 12, no. 3, pp. 325–338, 2005.

[31] N. Mladenović and N. Hansen, "Variable neighborhood search," *Computers and Operations Research*, vol. 24, pp. 1097–1100, 1997.

[32] P. Hansen and N. Mladenovic, *Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization*, edited by s. voss et al. ed. Kluwer Academic Publishers, Dordrecht, 1999, ch. An introduction to variable neighborhood search, pp. 433–458.

[33] M. W. Chase, D. E. Soltis, R. G. Olmstead, D. Morgan, D. H. Les, B. D. Mishler, M. R. Duvall, R. A. Price, H. G. Hills, Y.-L. Qiu, K. A. Kron, J. H. Rettig, E. Conti, J. D. Palmer, J. R. Manhart, K. J. Sytsma, H. J. Michaels, W. J. Kress, K. G. Karol, W. D. Clark, M. Hedren, B. S. Gaut, R. K. Jansen, K.-J. Kim, C. F. Wimpee, J. F. Smith, G. R. Furnier, S. H. Strauss, Q.-Y. Xiang, G. M. Plunkett, P. S. Soltis, S. M. Swensen, S. E. Williams, P. A. Gadek, C. J. Quinn, L. E. Eguiarte, E. Golenberg, G. H. Learn, S. W. Graham, S. C. H. Barrett, S. Dayanandan, and V. A. Albert, "Phylogenetics of seed plants: An analysis of nucleotide sequences from the plastid gene rbcl," *Annals of the Missouri Botanical Garden*, vol. 80, no. 3, pp. 528–580, 1993. [Online]. Available: http://spectrum.library.concordia.ca/6741/

[34] D. F. Robinson and L. R. Foulds, "Comparison of phylogenetic trees," *Mathematical Biosciences*, vol. 53, pp. 131–147, 1981.

[35] Y. Lin, V. Rajan, and B. M. E. Moret, "A metric for phylogenetic trees based on matching," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 4, pp. 1014–1022, 2012.