Progressive Tree Neighborhood applied to the Maximum Parsimony Problem

Adrien Goëffon, Jean-Michel Richer and Jin-Kao Hao
LERIA - University of Angers, 2 bd Lavoisier, 49045 Angers Cedex 01, France
{goeffon,richer,hao}@info.univ-angers.fr

Abstract— **The Maximum Parsimony problem aims at reconstructing a phylogenetic tree from DNA sequences while minimizing the number of genetic transformations. To solve this NP-complete problem, heuristic methods have been developed, often based on local search. In this article, we focus on the influence of the neighborhood relations. After analyzing the advantages and drawbacks of the well-known NNI, SPR and TBR neighborhoods, we introduce the concept of *Progressive Neighborhood* which consists in constraining progressively the size of the neighborhood as the search advances. We empirically show that applied to the Maximum Parsimony problem, this progressive neighborhood turns out to be more efficient and robust than the classic neighborhoods using a descent algorithm. Indeed, it allows to find better solutions with a smaller number of iterations or trees evaluated.**

Index Terms— **optimization, combinatorial algorithms, phylogeny reconstruction, maximum parsimony**

I. Introduction

Phylogeny can be defined as the reconstruction of the evolutionary history of a set of species nowadays identified by their nucleic acid (DNA) or amino acid (AA) sequences. The evolutionary relationships between species are represented by a tree which can be rooted or unrooted and whose branches are labelled with lengths that correspond to the evolutionary distance between species. Hillis in [22] identifies many applications of phylogeny reconstruction like genetic evolution, taxonomy and classification or virus detection.

Some methods attempt to reverse a model of evolution which embodies certain knowledge about the process of evolution that affects the molecular sequences. Models of evolution vary in their complexity and are defined by a set of parameters. For example, the Jukes Cantor (JC) model which considers that all sites evolve identically and independently requires just one parameter. Another model is the Kimura 2-Parameters model which is a generalization of the JC model.

It should be noticed that the assumptions taken into account for the search of an optimal tree of evolution may not reflect the reality of the evolution process. For example, the parsimony criterion considers that the most probable tree, or the best tree is the tree which minimizes the number of changes (or mutations), while it might not be the case for the natural evolution process (parsimony assumes that mutations are rare but this may not be the case). It is nevertheless interesting to rely on a mathematical model of a natural process in order to infer some new hypothesis that could be verified by further experiences.

In the past much work has been devoted to the problem of phylogeny reconstruction and we can bring three different approaches to light: *Distance methods*, inspired by the clustering work of Sokal and Sneath [34], introduced in 1967 by Cavalli-Sforza and Edwards [4] or by Fitch and Margoliash [12] rely on a matrix of distances observed between species. The most well-known algorithm of this class of methods is the *Neighbor-Joining* from Saitou and Nei [31], improved by Gascuel [15]. 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 Felsenstein [8] 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 [7] 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. However the search for an optimal tree is NP-complete as shown by Foulds and Graham [13]. 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 [16], [18], [28].

Among these heuristics, local search or neighborhood search, probably represents the most popular and effective method. Neighborhood search is based on an iterative improvement process using a given neighborhood relation which plays a key role in the effectiveness of neighborhood search methods.

In the literature, there exists three main neighborhoods for trees called NNI (of small size), SPR (medium) and TBR (large) whose properties make them more or less suitable to perform a superficial or a thorough search. In this paper we show how to combine the properties of those neighborhoods in order to improve the efficiency of local search algorithms. We introduce here a new neighborhood called *parametric progressive neighborhood* (PPN) for the resolution of the

MP problem [1]. The experimentations carried out on various benchmark instances show that a simple descent algorithm using PPN is able to converge rapidly toward high quality solutions.

The remaining of the paper is organized as follows. In the next section, we give a formal definition of the Maximum Parsimony Problem. In Section III, we describe three well-known tree neighborhoods commonly used by local search procedures and analyze their limits. We then propose in Section IV the framework of a *parametric Progressive Neighborhood*, which combines large and small neighborhoods properties. This progressive neighborhood is used within a descent algorithm and proves to be efficient compared to NNI, SPR or TBR as it allows to find good solutions in a small number of iterations. The next two sections are devoted to experimentations and comparisons. In the last two sections, we try to justify our approach and give some concluding remarks.

II. The Maximum Parsimony Problem

Given a multiple alignment of a set S of n species of length k nucleotides, the aim of the Maximum Parsimony problem is to find a phylogenetic tree 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 sequences of parsimony from the leaves to the root of the tree. More precisely we have the following definitions:

Definition 1 (Sequence of parsimony): Given two sequences S_1 and S_2 of length k 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 $\mathcal{P}(\Sigma)$, where $\Sigma = \{-, A, C, G, T\}$, the sequence of parsimony of S_1 and S_2, noted $F(S_1, S_2) =< z_1, \cdots, z_k >$ is obtained by (see Fitch [11]) :

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

The cost of the 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 (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:
  - the set of internal nodes I composed of $n - 1$ nodes that have 2 descendants, i.e. $I = \{w \in V / \exists u, v \in V \, such \, that \, \{(w, u), (w, v)\} \in E\}$,
  - the set of leafs L composed of n nodes which have no descendant, i.e $L = \{u \in V / \nexists v \in V, (u, v) \in E\}$,
- there exists an assignment ψ which is a bijection from the set of sequences S to the set of leafs L,

[1]The PPN program is available upon request from the authors of this paper.

- 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.

MP can also be formulated as a combinatorial optimization problem $(\mathcal{T}, \phi)$ as follows:

1) the search space $\mathcal{T}$ is defined by the set of all possible binary trees of parsimony, where $\mathcal{T}$ is of size $\prod_{i=3}^{|S|}(2i - 3)$ [32],
2) the cost function $\phi : \mathcal{T} \to \mathbb{N}$ is the score of parsimony of a tree.

Note that those definitions concern the Fitch parsimony, for which all changes have equal cost. Wagner or Camin-Sokal parsimony lead to different definitions. For a thorough introduction to the MP problem we refer the reader to [9].

III. Local search and neighborhoods

A. Iterative Descent

To solve the MP problem we have decided to use a local search algorithm based on an *iterative descent* (or *hill-climbing*) scheme [23] which is one of the most basic stochastic local search algorithms. The descent is guided by an evaluation (score) function ϕ and wanders through a *forest* of trees. Each tree is also called a *configuration* of the search space $\mathcal{T}$. Moreover, the search is strongly influenced by a neighborhood relation $\mathcal{N}$ which for a given tree T defines a subset of $\mathcal{T}$ called the neighbors of T.

The iterative descent starts from an initial, often randomly generated, configuration in the search space and tries to improve iteratively the current configuration with respect to ϕ. At each iteration, a neighboring configuration $T' \in \mathcal{N}(T)$ replaces the current configuration T, if T' has a better score, i.e. $\phi(T') < \phi(T)$.

The search ends when there is no more neighbor which can improve the score of the current configuration. The last configuration visited is called a solution of the problem.

Such a solution is also called a local optimum because the algorithm does not guarantee that it will end with the best solution of the problem, called a global optimum. Indeed stochastic local search algorithms mainly depend on the neighborhood relation used. Although we are not sure to obtain the best solution, local search proves to be a simple yet powerful framework which can provide good results.

B. NNI, SPR and TBR neighborhoods for trees

A neighborhood relation structures the search space through which a local search algorithm finds a path towards a solution. There are three well-known neighborhoods for trees in the literature: NNI, SPR and TBR.

NNI (*Nearest Neighbor Interchange*) [37] 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 [30].

SPR (*Subtree Pruning Regrafting*) [35] is a strategy which cuts a branch and reinserts it elsewhere. There are $2(n-3)(2n-7)$ possible SPR rearrangements [2] for each tree which makes it a medium size neighborhood.

TBR (*Tree-Bisection-Reconnection*) [35] consists in breaking the tree in two subtrees which will be reconnected from one of their branches. From a given tree, the TBR neighborhood induces at most $(2n-3)(n-3)^2$ neighbor trees [2]. TBR is thus a much larger neighborhood than NNI and SPR.

Finally, the following property can be verified [27]: NNI $\subseteq$ SPR $\subseteq$ TBR.

C. Properties and limits of known neighborhoods

A small neighborhood relation as NNI allows only local modifications on a tree topology, implying that a neighboring tree that results from such a transformation is very close to the initial tree. Consequently, the variation cost induced by a NNI transformation can be easily calculated. Another advantage of NNI is related to its small number of neighbors compared to other neighborhoods. The resolution time of a local search algorithm using NNI is then much smaller than with SPR or TBR. However, the drawback induced by NNI is that it has a weak capability to significantly improve the cost of a configuration after a few moves. And thus local optima far from the global optimum can sometimes be reached at an early stage of the search.

On the contrary a large neighborhood relation like TBR requires more computational effort. Exploring all the neighbors of a configuration using TBR is computationally expensive. Indeed, the neighboring trees are subject to important topological modifications. Thus, less information can be conserved for the calculation of the parsimony score of a neighbor tree even if Goloboff [18] proposed a method that aims at reducing the complexity of this calculation.

D. Escaping from local optima

To avoid being trapped in a local optima, existing methods propose different alternatives. For example, the method proposed by Nixon [28], often used in MP softwares, applies noising strategies [5] to modify the evaluation (cost) function when the local search procedure is stagnating. This enables one to perturb the current configuration while keeping moving through a search space of identical structure.

Another method used by Ribeiro *et al.* [1], [29] 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 SPR for which only leaves are pruned. This can be considered as an application of the VNS (Variable Neighborhood Search) metaheuristic, proposed by Hansen and Mladenovic [21]) to the MP problem. In [1], [29], Ribeiro *et al.* reported very good solutions for a set of benchmarks. For this reason, this method is used later in this paper as a reference for our experimental studies.

However, the interest of this VNS application seems limited. If the initial configuration is quite far from the optimum, the first local search step (with the most restricted neighborhood of the set) will quickly reach a local optimum. Even if one uses a large neighborhood in order to try to improve the quality of the current configuration, it is likely that a great part of the computational effort will be lost by considering more neighbors that will be shown to be useless, thus loosing a great part of the information gained during the first part of the local search.

IV. Progressive neighborhood

A. General Principle

In order to combine the interesting properties of large and small neighborhoods, we propose to start the search from a large neighborhood whose size decreases progressively as the search goes forward.

Contrary to VNS which enlarges its neighborhoods progressively, starting the search from a large neighborhood may prove to be a very interesting alternative. Indeed, a large neighborhood allows to take into account more neighbors (or topologies) and ensures a more intensive search. Using a large neighborhood at the early stage of the search allows to quickly improve the quality of the best configuration found, thus providing a good tree topology for the later stages of the search. Near the end of the search, the changes applied to the tree topology should be limited via a smaller neighborhood. In fact, important topological changes might not be relevant at late stages of the search and could be considered as starting a new search. Such a search scheme may be achieved by progressively reducing the size of the neighborhood.

The basic idea behind the progressive neighborhood search is thus to introduce a parametric neighborhood relation that evolves during the search, either statically according to a prefixed schedule or reactively according to some information collected during the search.

In the rest of this section, we define a simple progressive neighborhood search scheme that is based on two existing neighborhoods (SPR and NNI). We show how such a scheme can lead to high quality solutions for the MP problem with fast computing time. Later in Section VI, more empirical justifications are given to support the rationale of this progressive neighborhood.

B. Application to the MP problem

1) Insight of the method: As an illustration example, let us consider two neighborhoods $\mathcal{N}^1$ and $\mathcal{N}^2$ such that $\mathcal{N}^2 \subseteq \mathcal{N}^1$, in order to define a simple parametric neighborhood $\mathcal{N}_d$ that can range through $\mathcal{N}^1$ and $\mathcal{N}^2$. Consider $\mathcal{N}^1 = SPR$ and $\mathcal{N}^2 = NNI$. Note that NNI can be considered as a special case of SPR with the constraint that a pruned branch T must be reinserted to an edge adjacent to T in the initial tree. By extension, let us imagine a neighborhood of type SPR where the distance between the pruned edge and the edge receiving the branch insertion is constrained by a specific value d. If there is no limitation on the distance ($d = \infty$), we obtain the SPR neighborhood. If the distance is set to its minimum value

(d = 1), we obtain the NNI neighborhood (see Figures 1 and 2).

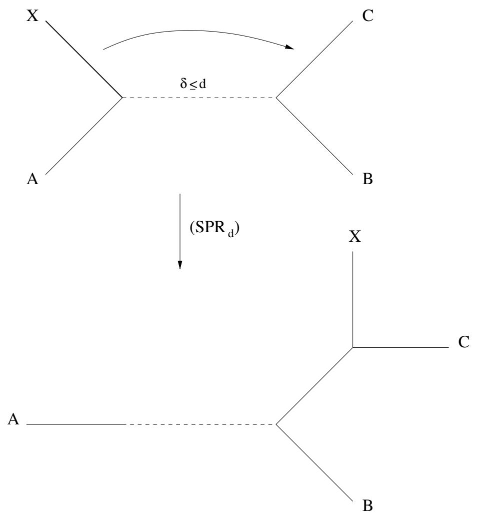

Fig. 1. SPR and parametric progressive neighborhood: SPR corresponds to the case where $d = \infty$

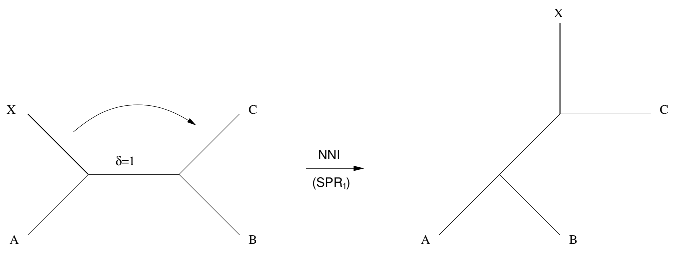

Fig. 2. NNI and parametric progressive neighborhood: NNI corresponds to the case where d = 1

2) A parametric neighborhood that collapses: In order to enable the neighborhood to progressively collapse we introduce the notion of distance (in terms of edges) between two nodes v_i and v_j. The node v_i is the root of a subtree which is pruned from the current tree, and that will be regrafted between v_j and its direct ancestor.

Let us now define more precisely the distance between v_i and v_j, noted by $\delta(v_i, v_j)$, as the length of the elementary path between the respective ascendants of v_i and v_j, minus 1 if the path contains the root (in the case of rooted trees). Thus, two nodes with the same direct ancestor have a distance of 0. Notice that the distance defined previously also applies to unrooted trees.

In order to control the size of the neighborhood during the search, we introduce a parameter d and define PPN as the set of configurations of SPR such that the distance between a pruned edge and its inserted edge (*i.e.* distance δ between their two descendant nodes) is at most equal to d.

To start with the the medium size SPR neighborhood and progressively end with the small NNI neighborhood, we compute the initial and final values of the parameter d as follows:

$$\left\{ \begin{array}{l} \mathcal{N}_{d_{init}} \equiv SPR \\ \\ \mathcal{N}_{d_{final}} \equiv NNI \end{array} \right. \Rightarrow \left(\begin{array}{c} d_{init} \\ d_{final} \end{array} \right) = \left(\begin{array}{c} \max_{V^2} \delta(v_i, v_j) \\ 1 \end{array} \right)$$

d_{init} depends on T_{init}, i.e. the tree that initiates the search, and corresponds to the smallest upper bound of distances between nodes, which is equal to the longest distance between all pairs of leaves.

If we linearly reduce d then the value of d at the i-th stage of a local search of a maximum of M iterations is equal to $\lfloor d_{init} \left(1 - \frac{i}{M}\right) \rfloor$ $(i < M)$.

In the following section we experimentally evaluate the impact of the progressive neighborhood PPN compared to NNI and SPR.

V. Experimentations

A. Benchmarks

In order to show the potential of the progressive neighborhood, we used a set of benchmarks composed of both random and real instances.

Random instances were generated with *dnatree* [25], using the Kimura 2-parameters model [24]. 300 instances generated according to different parameters, were used for the tests. Few variations among the different results were observed in regard to the nature of the instance. To lighten the reading, we have selected six instances of different sizes that are a representative sample of all tests performed. They contain 100, 300 or 500 DNA sequences, either short (100 nucleic acids) or long (1000). The transition / transversion rate is fixed to 2 and the mutation probability by time unit set to 5%. Instances are identified by two numbers: the number of sequences and their length (100-100, 300-100, 500-100, 100-1000, 300-1000, 500-1000).

We also used the real *Zilla* [6] instance, which is commonly cited in the literature. This instance is composed of 500 sequences of 759 nucleotides and has a best known score of parsimony of 16 218 [28] which is believed to be very difficult to reach.

An interesting set of instances from [29] was also taken into consideration in order to evaluate the ability of the PPN neighborhood to help the descent quickly converge to a near optimal solution.

B. Experimental conditions

The PPN neighborhood proposed in this paper is compared to the well-known neighborhoods NNI and SPR using an iterative descent algorithm. The only difference is that the descent can accept neighbors with the same score, in order to avoid being trapped in a local optimum.

Since $NNI \subseteq PPN \subseteq SPR$, the descent based on SPR is supposed to globally return better solutions. Indeed, a local optimum with respect to one neighborhood is also a local optimum with respect to all included neighborhoods.

But finding a true optimal solution involves looking through all the neighbors at each iteration, which may be computationally intractable for large instances and large neighborhoods.

For this reason, we have chosen to use a descent algorithm based on the *first improvement* principle. Therefore, instead of examining all the neighbors of a current tree at each iteration, the neighbors are examined in a fixed order so as to systematically find a neighbor with a score smaller or equal to the current tree. The descent procedure ends when it reaches a maximum of M iterations which is also the number of trees evaluated. The pseudo-code given in Algorithm 1 shows the skeleton of the descent procedure used to perform our tests.

Algorithm 1 Descent algorithm with a fixed number of iterations

```
Input: $n$ aligned sequences, $\mathcal{N}$ a neighborhood relation and $M$ the maximum number of iterations (or trees evaluated)
Output: The best tree found
Generate an initial tree $T \in \mathcal{T}$, randomly ($\mathcal{R}$) or with a greedy method ($\mathcal{G}$)
$nbIter = 0$
While $nbIter < M$ do
    for each $T' \in \mathcal{N}(T)$ do
        if $\phi(T') \leqslant \phi(T)$ then
            $T = T'$
            break
        $nbIter = nbIter + 1$
return $T$
```

In our experiments, we have set M to 50 000 for medium size random instances (100 or 300 sequences). For large instances ($\geqslant$ 500 sequences), M is set to a more important value (see Section V-C for details).

Table I gives an overview of the number of neighbors $|\mathcal{N}(T)|$ of a tree T depending on the number of sequences n (see Section III-B), for the NNI, SPR and TBR neighborhoods. Note that the number of unrooted trees with n leaves is equal to the number of rooted trees with $n-1$ leaves.

n	$\|NNI\|$	$\|SPR\|$	$\|TBR\|$	$\|\mathcal{T}\|$
50	94	8 792	$\leqslant 2.1 \times 10^5$	2.8×10^{76}
100	194	37 442	$\leqslant 1.8 \times 10^6$	3.3×10^{184}
300	594	352 242	$\leqslant 5.2 \times 10^7$	3.4×10^{700}
500	994	987 042	$\leqslant 2.5 \times 10^8$	1.0×10^{1280}

TABLE I
SIZES OF THE NEIGHBORHOODS NNI, SPR AND TBR AND TOTAL NUMBER OF NEIGHBORS FOR A TREE OF n LEAVES

For all instances, we have run the descent algorithm with each neighborhood (NNI, SPR and PPN) 100 times. In the following tables the different combinations appear as *Initial+Neighborhood* where *Initial* indicates the method used to build the first tree which initiates the descent and *Neighborhood* is the neighborhood used *i.e.* NNI, SPR or PPN within the local search framework of Algorithm 1. In order to compare the behavior of each descent for different initial solutions, we use a random ($\mathcal{R}$) and a greedy ($\mathcal{G}$) algorithm which are described in [1] as *first random step* and *first random operational taxon unit insertion over greedy branch*. Descents from random trees $\mathcal{R}$ assess the ability of the methods to converge to a near optimal solution. $\mathcal{G}$ is a greedy algorithm which builds good initial tree and may be helpful in order to find a better solution.

		ϕ_0	ϕ_b	$\mathfrak{f}$ (%)	ϕ_a	σ	*time*
100-100	$\mathcal{R}$+SPR	*1 570*	534	6	536.3	1.5	0.4
	$\mathcal{R}$+NNI		534	5	566.7	32.4	0.3
	$\mathcal{R}$+**PPN**		**534**	**87**	**534.4**	**2.3**	**0.3**
	$\mathcal{G}$+SPR	*544*	534	26	534.9	0.6	0.5
	$\mathcal{G}$+NNI		534	42	536.2	2.7	0.4
	$\mathcal{G}$+**PPN**		**534**	**82**	**534.2**	**0.4**	0.4
100-1000	$\mathcal{R}$+SPR	*12 914*	4 080	10	4 093.7	14.3	3.7
	$\mathcal{R}$+NNI		4 080	46	4 121.9	91.4	3.2
	$\mathcal{R}$+**PPN**		**4 080**	**100**	**4 080.0**	**0**	**2.2**
	$\mathcal{G}$+SPR	*4 085*	4 080	85	4 080.3	0.5	4.0
	$\mathcal{G}$+NNI		4 080	75	4 080.3	0.6	3.4
	$\mathcal{G}$+**PPN**		**4 080**	**100**	**4 080.0**	**0**	**2.8**

TABLE II
RESULTS OBTAINED WITH A DESCENT ALGORITHM ON SMALL SIZE RANDOM INSTANCES FOR THREE DIFFERENT NEIGHBORHOODS: NNI, SPR, PPN.

C. Results

Tables II, III and IV contain for each instance and each neigborhood the following data: the score of the initial tree ϕ_0, the best score (or best tree) obtained when the search ends ϕ_b and its frequency $\mathfrak{f}$, the average score of trees for a given number of runs ϕ_a, the standard deviation σ of ϕ_a and the average CPU $time$ in seconds for a given number of runs[2].

1) Random instances: For small random instances (see Table II), we notice that the descent algorithm with the progressive neighborhood (PPN) returns solutions with the best score. NNI does not seem to be reliable when the initial solution is randomly built.

For medium size instances (see Table III), we observed that the SPR neighborhood is not appropriate any more for short descents. The quality of solutions returned with NNI is strongly dependent on the initial tree and can converge prematurely to a local optimum of poor quality. This shows the instability of this method due to stochastic factors. The progressive neighborhood PPN, on the other hand, obtains good results in any case. On the 300-100 instance, we have tested a descent using the TBR neighborhood, but it seems not effective with those experimental conditions.

The efficiency of the neighborhoods can be better observed on Figure 3 which shows the evolution of the score of trees for a given number of iterations of the descent algorithm. We observe that the descent algorithm using the progressive neighborhood clearly dominates SPR and NNI.

For large instances (see Table IV, for which time is given in seconds), we observed that SPR generally yields bad results. Scores returned by NNI are subject to high variations with important standard deviations. Increasing the number of iterations (M) enables SPR to find better results (with a more important computation time) but still far from the best results of the

[2]Implemented in C++, all algorithms are compiled using the gcc/g++ compiler and run on a 2.4 GHz Opteron 850 processor.

		ϕ_0	M	ϕ_b	$\mathfrak{f}$ (%)	ϕ_a	σ	*time*
300-100	$\mathcal{R}$+TBR	*5 864*	5.10^4	1 601	1	1 671.3	36.2	10.5
	$\mathcal{R}$+SPR			1 583	1	1 649.5	30.3	0.8
	$\mathcal{R}$+NNI			1 569	1	1 954.3	151.6	0.4
	$\mathcal{R}$+**PPN**			**1 300**	**1**	**1 329.1**	**38.6**	**0.6**
	$\mathcal{R}$+SPR		5.10^5	1 301	1	1 307.3	3.2	6.5
	$\mathcal{R}$+NNI			1 329	1	1 632.4	143.2	5.7
	$\mathcal{R}$+**PPN**			**1 300**	**30**	**1 301.9**	**5.5**	**6.6**
	$\mathcal{G}$+SPR	*1 325*	5.10^4	1 307	2	1 315.4	4.3	1.1
	$\mathcal{G}$+NNI			1 300	1	1 304.7	2.7	0.8
	$\mathcal{G}$+**PPN**			**1 300**	**4**	**1 302.6**	**1.4**	**1.2**
	$\mathcal{G}$+TBR		5.10^5	1 303	3	1 307.4	2.1	121.8
	$\mathcal{G}$+SPR			1 300	1	1 302.8	1.4	7.0
	$\mathcal{G}$+NNI			1 301	10	1 304.2	2.7	5.8
	$\mathcal{G}$+**PPN**			**1 300**	**19**	**1 301.0**	**0.7**	**8.0**
300-1000	$\mathcal{R}$+SPR	*51 387*	5.10^4	16 789	1	17 422.8	253.1	9.2
	$\mathcal{R}$+NNI			14 209	15	14 381.8	213.8	5.8
	$\mathcal{R}$+**PPN**			**14 209**	**94**	**14 209.5**	**1.8**	**4.2**
	$\mathcal{R}$+SPR		5.10^5	14 209	1	14 234.7	16.4	64.2
	$\mathcal{R}$+NNI			14 209	33	14 352.6	202.2	51.3
	$\mathcal{R}$+**PPN**			**14 209**	**100**	**14 209.0**	**0**	**40.9**
	$\mathcal{G}$+SPR	*14 230*	5.10^4	14 209	1	14 221.4	8.0	22.0
	$\mathcal{G}$+NNI			14 209	76	14 209.7	1.9	15.1
	$\mathcal{G}$+**PPN**			**14 209**	**100**	**14 209.0**	**0**	**20.3**
	$\mathcal{G}$+SPR		5.10^5	14 209	87	14 209.3	0.8	76.2
	$\mathcal{G}$+NNI			14 209	78	14 209.5	1.8	59.7
	$\mathcal{G}$+**PPN**			**14 209**	**100**	**14 209.0**	**0**	**60.0**

TABLE III

RESULTS OBTAINED WITH A DESCENT ALGORITHM ON MEDIUM SIZE RANDOM INSTANCES FOR THE NEIGHBORHOODS: NNI, SPR, TBR, PPN.

		ϕ_0	M	ϕ_b	$\mathfrak{f}$ (%)	ϕ_a	σ	*time*
500-100	$\mathcal{R}$+SPR	*9 164*	5.10^4	3 493	1	3 671.6	68.6	1.1
	$\mathcal{R}$+NNI			4 039	1	4 410.9	174.7	0.9
	$\mathcal{R}$+**PPN**			**2 279**	**1**	**2 367.4**	**53.2**	**1.1**
	$\mathcal{R}$+SPR		5.10^5	2 312	1	2 344.7	15.7	8.5
	$\mathcal{R}$+NNI			2 263	1	2 428.2	108.2	6.5
	$\mathcal{R}$+**PPN**			**2 243**	**48**	**2 244.0**	**3.0**	**9.8**
	$\mathcal{G}$+SPR	*2 265*	5.10^4	2 259	1	2 273.1	6.7	2.6
	$\mathcal{G}$+NNI			2 243	7	2 247.2	3.6	1.6
	$\mathcal{G}$+**PPN**			**2 243**	**4**	**2 245.3**	**1.5**	**3.4**
	$\mathcal{G}$+SPR		5.10^5	2 246	2	2 251.3	2.8	9.1
	$\mathcal{G}$+NNI			2 243	28	2 245.6	3.0	7.2
	$\mathcal{G}$+**PPN**			**2 243**	**61**	**2 243.5**	**0.7**	**13.1**
500-1000	$\mathcal{R}$+SPR	*100 388*	5.10^4	35 980	1	37 690.7	580.4	17.6
	$\mathcal{R}$+NNI			25 821	1	29 080.7	1767.8	18.3
	$\mathcal{R}$+**PPN**			**24 319**	**5**	**24 391.9**	**141.6**	**5.2**
	$\mathcal{R}$+SPR		5.10^5	24 938	1	25 205.6	116.0	100.4
	$\mathcal{R}$+NNI			24 319	3	24 618.5	593.2	66.3
	$\mathcal{R}$+**PPN**			**24 319**	**100**	**24 319.0**	**0**	**54.2**
	$\mathcal{G}$+SPR	*24 359*	5.10^4	24 326	1	24 351.4	13.6	85.2
	$\mathcal{G}$+NNI			24 319	70	24 321.0	3.4	49.4
	$\mathcal{G}$+**PPN**			**24 319**	**98**	**24 219.1**	**0.7**	**81.1**
	$\mathcal{G}$+SPR		5.10^5	24 319	6	24 325.8	5.3	160.1
	$\mathcal{G}$+NNI			24 319	76	24 320.6	3.1	104.0
	$\mathcal{G}$+**PPN**			**24 319**	**100**	**24 219.0**	**0**	**143.8**

TABLE IV

RESULTS OBTAINED WITH A DESCENT ALGORITHM ON LARGE SIZE RANDOM INSTANCES FOR THREE DIFFERENT NEIGHBORHOODS: NNI, SPR, PPN.

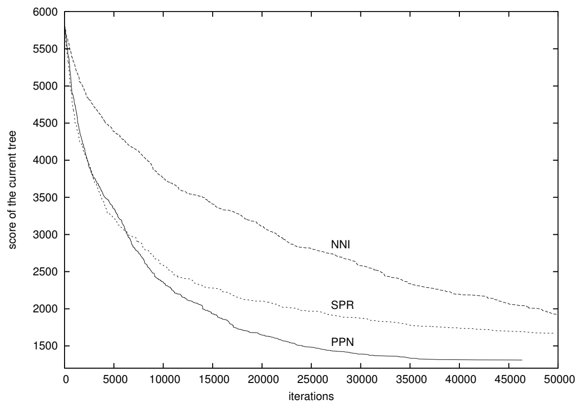

Fig. 3. Evolution of the score of trees for SPR, NNI and PPN with a 300-100 random instance starting from a random tree

		ϕ_0	M	ϕ_b	$\mathfrak{f}$ (%)	ϕ_a	σ	*time*
zilla	$\mathcal{R}$+SPR	*37 427*	10^5	18 013	1	18 470.2	126.9	18.5
	$\mathcal{R}$+NNI			19 875	1	21 863.2	894.9	10.6
	$\mathcal{R}$+**PPN**			**16 263**	**1**	**16 553.0**	**204.3**	**7.7**
	$\mathcal{R}$+SPR		10^6	16 272	1	16 312.8	20.4	135.3
	$\mathcal{R}$+NNI			19 142	1	21 342.4	1 102.9	118.4
	$\mathcal{R}$+**PPN**			**16 222**	**3**	**16 257.0**	**45.3**	**78.5**
	$\mathcal{R}$+SPR		2.10^7	16 219	5	16 225.8	4.8	2 325
	$\mathcal{R}$+**PPN**			**16 218**	**5**	**16 223.3**	**3.9**	**1 609**
	$\mathcal{G}$+SPR	*16 475*	10^5	16 335	1	16 384.1	21.9	69.1
	$\mathcal{G}$+NNI			16 255	1	16 313.0	22.4	39.6
	$\mathcal{G}$+**PPN**			**16 226**	**1**	**16 245.9**	**11.8**	**66.8**
	$\mathcal{G}$+SPR		10^6	16 237	1	16 256.4	8.3	203.8
	$\mathcal{G}$+NNI			16 257	1	16 311.3	28.1	141.3
	$\mathcal{G}$+**PPN**			**16 219**	1	**16 228.5**	**6.7**	**140.9**

TABLE V

RESULTS ON THE *Zilla* INSTANCE

other methods. Also note the efficiency of the method using the progressive neighborhood PPN for large instances like the 500-1000 instance. In 100 000 iterations, PPN succeeds to systematically return the best solution of score 24 319 with half of the time required by SPR. Among the 20 runs performed, NNI has found this score only once.

2) Real instances: For the well-known *Zilla* instance, it is believed that it is very difficult to find the best known score of 16 218 (see [28]). This was confirmed during the experimentations we carried out.

We have performed different tests with an increasing number of iterations ($M = 10^5$ and 10^6) and results of the three methods are reported in Table V. NNI systematically returns poor solutions from random trees in less than 100 000 iterations and won't improve for a greater value of M. On this difficult instance, PPN obtains better scores than SPR. Moreover, we have run SPR and PPN from random trees 20 times for $M = 2 \times 10^7$ iterations. PPN could find the best known score within 30 minutes. Although the algorithm is quite simple, this result may be considered very interesting.

Figure 4 shows the evolution of the score for the three neighborhoods used to solve the *Zilla* instance within 300 000 iterations and from a tree built with the UPGMA distance-based method [33]. With this intermediate and quite good

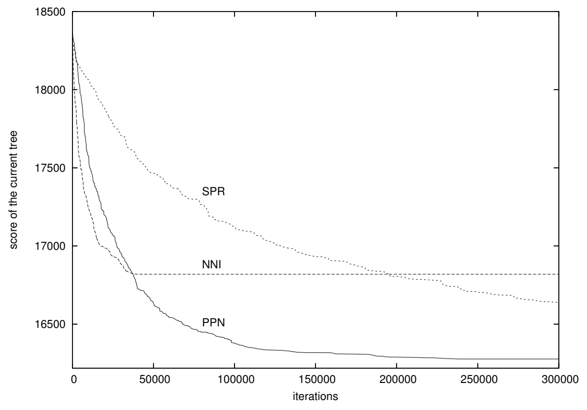

Fig. 4. Evolution of the score from an intermediate solution on the *Zilla* instance

tree, we can see that NNI is more effective than SPR at the beginning of the search before reaching local optima, contrary to the case for which the descent started from initial trees randomly generated (Figure 3). This is a first obervation which justifies the use of a progressive neighborhood.

D. Comparison with other methods and softwares

We would like to underline that the aim of this work is not the development of a new software able to compete with the best softwares like TNT (see section V-F) because they make an intensive use of several heuristics and strategies. Our aim here is to present a new neighborhood which, used with a simple descent algorithm, turns out to be more efficient than the existing neighborhoods generally used for the resolution of the Maximum Parsimony problem. We believe that this new neighborhood could be used to improve the efficiency of the existing softwares. In the following, we won't try to compare the resolution time of each method or software but some of them are reported in order to give an overview of the effort that is required to solve a particular problem.

E. Comparison with GRASP+VND

Finally, we assess the performance of PPN on a set of 8 real-life instances (see Table VI) taken from [29], which are initially presented in [26]. Table VI reports the best known scores obtained by a GRASP+VND procedure [29], which could find three new optima and could redicover the other known values. The authors recall that the previous best known solutions were not found with their method. For each instance, the column named *time* corresponds to the computation time (in seconds) used to obtain the best score out of 10 executions of the GRASP+VND algorithm on a 2GHz Pentium IV processor. This information is provided for indication purpose only and may give some evidence about the difficulty of each problem instance. For example, one may notice from Table VI that the last four instances seem more difficult given that much more computing times are needed to reach the best scores by GRASP+VND.

GRASP+VND is a very recent procedure which applies two well-known metaheuristics GRASP [10] and VND [21]

			GRASP+VND	
Instance	n	k	ϕ_b	*time*
GRIS	47	93	**172**	3 505
ANGI	49	58	**216**	5 099
TENU	56	179	**682**	7 497
ETHE	58	86	**372**	10 042
ROPA	75	82	**325**	15 764
GOLO	77	97	**496**	32 836
SCHU	113	146	**759**	113 391
CARP	117	110	**548**	82 176

TABLE VI
SIZE AND BEST-KNOWN SCORES ON INSTANCES TAKEN FROM [29]

	PPN					
Instance	ϕ_b	M	$\mathfrak{f}$ (%)	ϕ_a	σ	*time*
GRIS	**172**	5×10^4	100	172.0	0	1.7
ANGI	**216**		8	218.0	1.4	1.3
TENU	**682**		9	686.6	3.7	2.6
ETHE	**372**		2	374.3	1.0	1.5
ROPA	326		5	328.7	1.6	1.6
GOLO	497		1	505.8	4.2	2.1
SCHU	**759**		1	768.6	5.1	3.5
CARP	549		1	554.8	3.1	3.0
ROPA	**325**	2×10^5	3	328.1	1.7	7.3
GOLO	**496**		2	502.7	2.9	8.0
SCHU	**759**		10	766.4	4.5	12.4
CARP	**548**		1	553.5	3.5	10.9

TABLE VII
PERFORMANCES OF PPN ON BENCHMARKS TAKEN FROM [29]

to the MP problem. An initial solution is built with a greedy randomized algorithm and is then improved with a VND heuristic using the Multiple Subtree Pruning and Regrafting neighborhood (l-SPR). Note that Ganapathy, Ramachandran and Warnow define in [14] the p-Edge-Contract-and-Refine (p-ECR) neighborhood which can be seen as a similar extension of NNI. In practice, l and p are limited to 2 for those multiple steps neighborhoods.

For PPN, we have limited the number of iterations of the descent to $M = 50\ 000$. We have executed the PPN procedure 100 times on each instance (on a 2.4GHz AMD Opteron processor) and report in Table VI the value of the best solution found (ϕ_b), the average computation time in seconds (time) as well as the frequency of the best solution found ($\mathfrak{f}$). Moreover, we indicate for each instance the average score (ϕ_a) and its standard deviation (σ).

From Table VII, one observes that within $M = 50\ 000$ iterations, PPN can find the optimum score for 5 out of 8 instances in a very short time (a few seconds). If we extend the search to $M = 200\ 000$ for the 4 most difficult instances, PPN is able to find all the best known results. Those results prove the efficiency of PPN due to the limited number of trees visited during the search.

F. Comparison with TNT

The software TNT [20] (for Tree analysis using New Technology) is probably the fastest and most effective parsimony analysis program. TNT uses many search strategies such as

	TNT											
	PPN search			SPR search			TBR search			New Tech. search		
Instance	ϕ_b	ϕ_a	*trees*	ϕ_b	ϕ_a	*trees*	ϕ_b	ϕ_a	*trees*	ϕ_b	ϕ_a	*trees*
GRIS	**172**	172.0	5.0×10^4	**172**	173.7	1.1×10^5	172	172.1	4.3×10^5	**172**	172.0	5.7×10^6
ANGI	**216**	218.0	5.0×10^4	**216**	219.2	1.8×10^5	216	218.6	7.7×10^5	**216**	216.0	8.6×10^6
TENU	**682**	686.6	5.0×10^4	**682**	688.0	1.9×10^5	**682**	683.5	8.1×10^5	**682**	682.0	1.5×10^7
ETHE	**372**	374.3	5.0×10^4	**372**	376.8	2.3×10^5	**372**	374.1	9.0×10^5	**372**	372.2	1.0×10^7
ROPA	**325**	328.1	2.0×10^5	327	329.6	4.0×10^5	326	328.3	1.8×10^6	**325**	325.0	2.3×10^7
GOLO	**496**	502.7	2.0×10^5	498	507.2	5.5×10^5	497	502.7	2.6×10^6	**496**	496.3	2.3×10^7
SCHU	**759**	766.4	2.0×10^5	761	771.1	1.3×10^6	**759**	762.8	7.8×10^6	**759**	759.0	6.8×10^7
CARP	**548**	553.5	2.0×10^5	**548**	553.6	1.4×10^6	**548**	553.1	7.3×10^6	**548**	548.0	6.1×10^7

TABLE VIII

COMPARISON OF PPN AND TNT USING TRADITIONAL SEARCH (SPR, TBR) OR NEW TECHNOLOGY SEARCH

tree drifting, parsimony ratchet or sectorial search. TNT has also the ability to evaluate millions of trees in a very efficient way [18]. Table VIII reports the number of trees evaluated during a SPR, TBR and a New Technology search. We can observe that the 4 methods presented (PPN vs TNT) show their ability to reach an optimal or a near optimal solution for each instance but PPN evaluates less trees than other methods especially on hard instances as reported on the column *trees* which represents the number of trees evaluated during the search.

G. Discussion

The reduction scheme of the progressive neighborhood implies that a maximum number of iterations M has to be fixed. The bigger M is, the slower the size of the neighborhood will decrease. If M is too big, SPR and PPN have the same behaviour. Indeed, the first third of the iterations of PPN uses a neighborhood close to SPR. If M is set to a small value, PPN will obtain good local optima but with less effort than SPR.

VI. JUSTIFICATIONS

In order to justify the validity of the progressive neighborhood, we have investigated the behavior of a descent algorithm with a SPR neighborhood and its influence on the search of a better configuration. In this section, we use δ to note the distance $\delta(v_i, v_j)$ between a tree T and a neighbor in SPR.

The first study aims at observing the evolution of the distance δ between a given tree T and a *best* neighbor T' which is one the best neighbors of T. For this purpose, we have used a *best improvement* descent algorithm (see algorithm 2). At each iteration of the algorithm, a best improving neighbor is chosen to replace the current tree. Recall that an improving neighbor has a better (smaller) parsimony score than the current tree. The algorithm stops when there is no more improving neighbor.

We have run this algorithm several times ($R = 10$) on 6 different random instances. Figure 5 shows the results obtained on an instance of 100 sequences of 1000 characters. The black line on Figure 5 is a polynomial approximation of degree 6 of the points.

From Figure 5, one observes clearly that at the beginning of the search, the distance δ is quite important. This distance

Algorithm 2 *Best improvement descent* using a SPR neighborhood which returns the best tree in function of the tree quality

```
Repeat $R$ times
    Generate an initial random tree $T_0$
    $i = 0$
    While $T_i$ is not a local optimum do
        find $T'$ on of the best neighbor of $T$
        $i = i + 1$
    $p = i$ the final number of iterations
    Print coordinates $(\frac{i}{p}, \delta(T_{i-1}, T_i))$, $\forall i \in \{1..p\}$
```

decreases when the search progresses. Near the end of the search, the current tree and its best neighbor are quite close, separated only by a small distance δ near 1.

We can say that to obtain a best improving neighbor at the early stage of the search, it is necessary to make important tree transformations, i.e. using a large neighborhood such as SPR or TBR. Since the distance between a tree and its best neighbor decreases along with the progress of the search, limited tree transformations will be sufficient near the end and a small neighborhood like NNI seems appropriate.

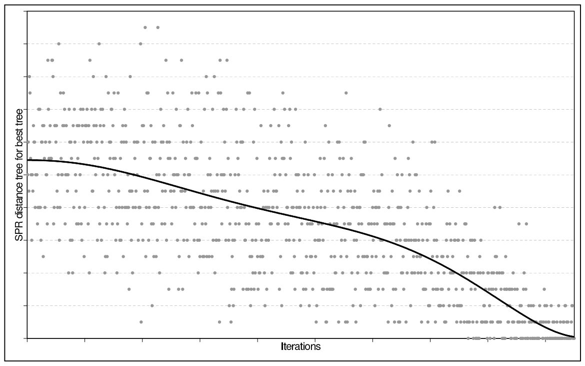

Fig. 5. Evolution of the distance δ between a tree and its best neighbor according to the SPR transformation in function of the number of iterations.

The second study we have carried out aims at evaluating the evolution of the number of improving neighbors (i.e. trees having a smaller parsimony score than the current tree). In order to correlate this information with the one related to the distance δ, we consider, for a given distance of δ, the evolution of the ratio :

$$\frac{number\,of\,improving\,neighbors}{total\,number\,of\,neighbors}$$

during a search (see Figure 6).

We can notice from Figure 6 that as long as the search progresses, the number of improving neighbors decreases and that the improving neighbors are more and more close (in terms of distance δ) to the current tree. Towards the end of the search, there are almost no more improving trees with large values of δ. This observation constitutes therefore another empirical justification for the use of a progressive neighborhood.

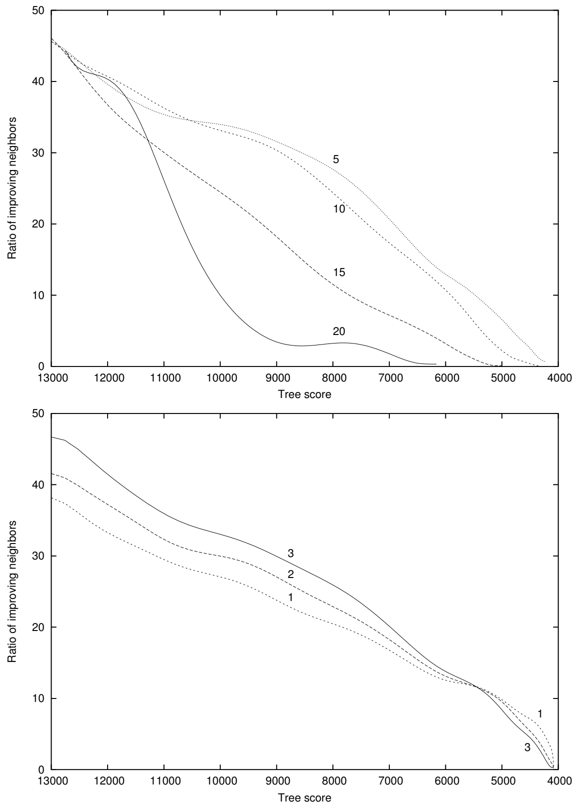

Fig. 6. Evolution of the ratio number of improving neighbors /total number of neighbors in function of the tree score, for a given distance δ (20, 15, 10, 5, 3, 2, 1).

VII. Conclusion

The local search heuristics using the SPR and NNI neighborhoods are among the most popular search methods to solve the MP problem in phylogenetic reconstruction. Although highly effective and fast for small instances (consisting of less than 100 species), they may not be reliable and efficient enough when applied to larger instances. In the case of a small size neighborhood like NNI, several searches have to be performed from distinct trees (replications) in order to explore several areas of the search space and thus to avoid the local optima pitfalls. In the case of larger neighborhoods like SPR or TBR, the search space is so huge that it is necessary to use a combination of methods (genetic algorithms [19], supertrees [3], ...) in order to obtain results of acceptable quality.

It would be also interesting to carry out a thorough study using TBR. In a first approximation TBR seems too large to provide any improvement compared to SPR, but this might be due to the linear reduction scheme which might not be appropriate.

The aim of this study was to understand the efficiency of the well-known neighborhoods and to propose an alternative that combines the interesting properties of the SPR and NNI neighborhoods. The *progressive neighborhood* (PN) introduced in this article shows a significant improvement compared to SPR and NNI, especially with difficult instances. Moreover, the progressive neighborhood is much more robust as it allows to minimize the replication of runs. Another interesting point is that the quality of the first tree used to begin the search does not seem to have much influence on the quality of the solution when using a descent algorithm with PN.

Finally, given the characteristics of the progressive neighborhood, it would be particularly interesting to integrate this neighborhood within the best current phylogeny reconstruction softwares like PAUP* [36] or TNT [20] in order to improve their efficiency.

Indeed, we have designed a hybrid algorithm called Hydra (for HYbrid Distance Recombination Algorithm) that combines a genetic algorithm using a new crossover operator DiBIP and a progressive neighborhood based local search algorithm [17]. The experimentations we have carried out on a set of 28 (real and randomly generated) benchmark instances show very competitive results compared to state-of-the-art algorithms. Indeed, for the 8 real datasets, the best known score are systematically found. The most remarkable results concern the set of 20 random instances for which we can improve 19 of the best scores known to day.

Acknowledgments: This work is partially supported by the French Ouest Genopole®. We would like to thank Dr. Olaf R.P. Bininda-Edmonds who provides us with the *Zilla* instance and Pr. Celso C. Ribeiro who has given us the instances of [29]. We also would like to thank the reviewers for their wise comments which helped us improve this article.

References

[1] A. A. Andreatta and C. C. Ribeiro. Heuristics for the phylogeny problem. *Journal of Heuristics* 8:429-447, 2002.
[2] B. L. Allen, M. Steel. Subtree Transfer Operations and their Induced Metrics on Evolutionary Trees. *Annals of Combinatorics* 5(1):1-15, 2000.
[3] O. R. P. Bininda-Emonds, Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life. *Computational Biology*, volume 4. Kluwer Academic Publishers, Dordrecht, the Netherlands
[4] L. L. Cavalli-Sforza and A. W. F. Edwards. Phylogenetic analysis: models and estimation procedures. *Evolution* 32: 550-570, 1967.
[5] I. Charon and O. Hudry, The noising methods : A generalization of some metaheuristics. *European Journal of Operational Research*, 135 (2001) 86-101
[6] M. W. Chase *et al.* Phylogenetics of seed plants: an analysis of nucleotide-sequences from the plastid gene rbcL. *Annals of the Missouri Botanical Garden*, 80:528-580, 1993.
[7] A. W. F. Edwards and L. L. Cavalli-Sforza. The reconstruction of evolution. *Annals of Human Genetics* 27: 105-106, 1963.
[8] J. Felsenstein. Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution* 17: 368-376, 1981.

[9] J. Felsenstein. Inferring phylogenies. *Sinauer Associates*, 2003.
[10] T. A. Feo and M. G. C. Resende. Greedy randomized adaptative search procedures. *Journal of Global Optimization* 6:109-133, 1995.
[11] W. Fitch. Towards defining course of evolution: minimum change for a specified tree topology. *Systematic Zoology* 20:406-416, 1971.
[12] W. M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science* 155: 279-284, 1967.
[13] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics* 3:43-49, 1982.
[14] G. Ganapathy, V. Ramachandran et T. Warnow. On contract-and-refine transformations between phylogenetic trees. SODA 2004: 900-909, 2004.
[15] O. Gascuel. On the optimization principle in phylogenetic analysis and the minimum evolution criterion. *Biology and Evolution* 17:401-405, 2000.
[16] A. Goëffon, J.-M. Richer and J.-K. Hao. Local search for the maximum parsimony problem. *Lecture Notes in Computer Science* 3612: 678-683, Springer, 2005.
[17] 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* 4193: 761-770, Springer-Verlag, 2006.
[18] P. A. Goloboff. Character optimisation and calculation of tree lengths. *Cladistics* 9: 433-436, 1993.
[19] P. A. Goloboff. Analyzing Large Data Sets in Reasonable Times: Solutions for Composite Optima. *Cladistics* 15:415-428, 1999.
[20] P. A. Goloboff, J. S. Farris et K. Nixon. TNT: Tree Analysis Using New Technology, 2003.
[21] P. Hansen et N. Mladenovic. An introduction to Variable neighborhood search. Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization. Edited by S. Voss et al., 433-458, Kluwer Academic Publishers, Dordrecht, 1999.
[22] D. Hillis, C. Moritz and B. Mable. Molecular Systematics, Sinauer, Boston, 1996.
[23] H. H. Hoos and T. Stützle. Stochastic Local Search: Foundations and Applications. *Morgan Kaufmann publishers*, 2005.
[24] M. Kimura. A simple model for estimating evolutionary rates of base of base substitutions through comparative studies of nucleotide sequence. *Journal of Molecular Evolution* 16:111-120, 1980.
[25] M. K. Kuhner and J. Felsenstein. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, 11:459-468, 1994 (Erratum 12:525).
[26] M. Luckow and R. A. Pimentel. An empirical comparison of numerical Wagner computer programs. *Cladistics* 1:47-66, 1985.
[27] D. R. Maddison. The discovery and importance of multiple islands of most-parsimonious trees. *Syst. Zool.* 43(3):315-328, 1991.
[28] K. C. Nixon. The Parsimony Ratchet, a New Method for Rapid Parsimony Analysis. *Cladistics* 15:407-414, 1999.
[29] 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.
[30] D. F. Robinson. Comparison of labeled trees with valency three. *J. Combin. Theory*, 11:105-119, 1971.
[31] N. Saitou and M. Nei. The neighbor-Joining Method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406-425, 1987.
[32] E. Schröder. Vier Kombinatorische Probleme. *Z. Math. Phys.* 15:361-376, 1870.
[33] R. R. Sokal et C. D. Michener. A statistical method for evaluating systemaic relationships *University of Kansas Science Bulletin* 38:1409-1438, 1958.
[34] R. R. Sokal et P. H. A. Sneath. Principles of Numerical Taxonomy. W. H. Freeman, San Francisco, 1963.
[35] D. L. Swofford, G. J. Olsen. in D.M. Hillis and C. Moritz (Ed.) Phylogeny Reconstruction. *Molecular Systematics*, chapter 11:411-501, 1990.
[36] D. L. Swofford. PAUP*: Phylogenetic Analysis Using Parsimony 4.0 Beta, 2002. *Molecular Systematics*, chapter 11:411-501, 1990.
[37] M. S. Waterman and T. F. Smith. On the similarity of dendograms. *Journal of Theoretical Biology* 73:789-800, 1978.

Jin-Kao Hao holds a full Professor position in the Computer Science Department of the University of Angers (France) since 1999. He graduated from the School of Computer Science of the National University of Defence Technology (China) in 1982. He received his M.S., Ph.D. and Professorship Diploma HDR respectively in 1987, 1991, 1998 from the National Institute of Applied Sciences (Lyon), the University of Burgundy (Dijon) and the University of Montpellier II in France. His research lies in the design of effective heuristic and metaheuristic algorithms for solving large-scale combinatorial problems in various application areas. His current research interests in bioinformatics include microarray data classification, phylogenetic tree reconstruction and multiple sequence alignment.

Jean-Michel Richer received a PhD degree in Computer Science from the University of Burgundy (Dijon), France in 1999. After working as a senior developer for Simplex Solutions SA, he joined Pr Jin-Kao Hao's team at the University of Angers as an assistant professor in 2000. His current research interests concern computational biology and methods related to the resolution of combinatorial optimization problems.

Adrien Goëffon received a PhD degree in 2006 in computer science from the University of Angers, France. His research interests concern local search heuristics, evolutionary algorithms, and their application on large-scale combinatorial problems and bioinformatics.