The Gene-Duplication Problem: Near-Linear Time Algorithms for NNI Based Local Searches

Mukul S. Bansal and Oliver Eulenstein

Department of Computer Science, Iowa State University, USA
{bansal, oeulenst}@cs.iastate.edu

Abstract. The gene-duplication problem is to infer a species supertree from a collection of gene trees that are confounded by complex histories of gene duplication events. This problem is NP-complete and thus requires efficient and effective heuristics. Existing heuristics perform a stepwise search of the tree space, where each step is guided by an exact solution to an instance of a local search problem. A classical local search problem is the NNI *search problem*, which is based on the nearest neighbor interchange operation. In this work we (i) provide a novel near-linear time algorithm for the NNI search problem, (ii) introduce extensions that significantly enlarge the search space of the NNI search problem, and (iii) present algorithms for these extended versions that are asymptotically just as efficient as our algorithm for the NNI search problem. The substantially extended NNI search problem, along with the exceptional speed-up achieved, make the gene-duplication problem more tractable for large-scale phylogenetic analyses.

1 Introduction

The rapidly increasing amount of available genomic sequence data provides an abundance of potential information for phylogenetic analyses. Most phylogenetic analyses combine genes from presumably orthologous loci, or loci whose homology is the result of speciation. These analyses largely neglect the vast amounts of sequence data from gene families, in which complex evolutionary processes such as gene duplication and loss, recombination, and horizontal transfer generate gene trees that differ from species trees. One approach to utilize the data from such gene trees (gene families) is to reconcile the gene trees with species trees based on the duplication optimality criterion that was introduced by Goodman et al. [13]. The corresponding optimization problem is called the *gene-duplication* problem [15]. This problem can be viewed as a *supertree problem*, that is, assembling from a collection of input trees (the gene trees) a species supertree that contains all species found in at least one of the input trees. The decision version of the gene-duplication problem is NP-complete [17]. Existing heuristics aimed at solving the gene-duplication problem search the space of all possible supertrees guided by a series of exact solutions to instances of a local search problem [20]. The local search problem is to find an optimal phylogenetic tree under the duplication optimality criterion in the neighborhood of a given tree. The neighborhood is the set of all phylogenetic trees into which the given tree can be transformed by applying a tree edit operation. A variety of different tree edit operations have been discussed in the literature [24, 26], and in practice the rooted nearest neighbor interchange (NNI) tree edit

operation has shown much potential for phylogenetic studies [15, 22]. However, despite this potential, algorithms for local search problems based on NNI operations are still in their infancy. To conduct large-scale phylogenetic analyses, there is much need for more effective NNI based local search problems that can be solved efficiently.

In this work we extend the NNI neighborhood to the k–NNI neighborhood. The k–NNI neighborhood contains all trees that can be obtained by performing at most k successive NNI operations on the given tree.

Recently, efficient solutions were given for local search problems based on the standard SPR [2] and TBR [3] edit operations. It can be easily shown [11, 12] that 2 and 3–NNI neighborhoods of a tree have very small overlap with its SPR and TBR neighborhoods. This results in novel and potentially more effective local searches. We greatly improve on the complexity of the best known (brute-force) solutions for 2 and 3–NNI based local search problems. Furthermore, we show that each subsequent instance of the local search problem for 1, 2, and 3–NNI neighborhoods can be solved in linear time after the first instance is solved. This is especially desirable since standard local search heuristics for the gene-duplication problem typically involve solving several thousand instances of the local search problem. Our novel near-linear time algorithms provide much potential for making the gene-duplication problem more suitable for large-scale phylogenetic analyses.

1.1 Previous Results

The gene-duplication problem is based on the Gene Duplication model from Goodman et al. [13]. In the following, we (i) describe the Gene Duplication model, (ii) formulate the gene-duplication problem, and (iii) describe a heuristic approach of choice [20] to solve the gene-duplication problem.

Gene Duplication model The Gene Duplication model is well studied [19, 15, 18, 29, 7, 5, 14] and explains incompatibilities between a pair of "comparable" gene and species trees through gene duplications. A gene and a species tree are *comparable*, if a *leaf-mapping* exists that provides a leaf to leaf mapping that maps every gene to the species from which it was sampled. The minimum number of gene duplications that are necessary under the Gene Duplication model to explain the incompatibilities can be inferred from the mapping M, which is an extension of the given leaf-mapping. M maps every gene in the gene tree to the most recent species in the species tree that could have contained the gene. More precisely, M maps each gene to the least common ancestor of the species from which the leaves (genes) of the subtree rooted at the gene were sampled (given by the leaf-mapping). A gene in the gene tree is a *duplication* if it has a child with the same M mapping. The *reconciliation cost* for a gene tree and a comparable species tree is measured in the number of gene duplications in the gene tree induced by the species tree. The *reconciliation cost* for a given collection of gene trees and a species tree is the sum of the reconciliation costs for each gene tree in the collection and the species tree. The mapping function is linear time computable on a PRAM [29] through a reduction from the least common ancestor problem [4]. Hence, the reconciliation cost for a collection of gene trees and a species tree is computable in linear time.

Gene-duplication problem and heuristics The *gene-duplication problem* is to find for a given set of gene trees a comparable species tree with minimum reconciliation cost. This approach has been successfully applied to phylogenetic inference in snakes [27], vertebrates [21, 23], *Drosophila* [8], and plants [25] among others. However, the decision variant of this problem and some of its characterizations are NP-complete [17, 10] while some parameterizations are fixed parameter tractable [28, 16]. Therefore, in practice, heuristics (e.g. [20]) are commonly used for the gene-duplication problem, even though they are unable to guarantee an optimal solution. In these heuristics, a *tree graph* (see [1, 26]) is defined for the given set of gene trees and some fixed tree edit operation. Each node in the tree graph represents a unique species tree comparable with the given gene trees. An edge is drawn between two nodes exactly if the corresponding trees can be transformed into each other by one tree edit operation. The *reconciliation cost* of a node in the graph is the reconciliation cost of the species tree represented by that node and the given gene trees. Given an initial node in the tree graph, the heuristic's task is to find a maximal-length path of steepest descent in the reconciliation cost of its nodes and to return the last node on such a path. This path is found by solving the *local search problem* for every node along the path. The local search problem is to find a node with the minimum reconciliation cost in the neighborhood of a given node. The time complexity of the local search problem depends on the tree edit operation used.

Here, the edit operation of interest is the NNI operation [1, 6]. Rooted and unrooted NNI operations have been extensively studied [9]. An NNI operation on a species tree S (represented as an undirected graph) can be performed by "swapping" two of its node disjoint subtrees whose root nodes are connected by a simple path of length 3. The resulting tree graph is connected and every node has a degree of $\Theta(n)$, where n is the number of leaves in S. Thus, the local search problem for the k–NNI neighborhood and r gene trees can be solved naively in $O(rn^{k+1})$ time (assuming, for convenience, that the gene trees differ in size from the species tree by at most a constant factor). These brute-force solutions are the best available for $k \geq 1$, and hence, the development of faster algorithms is required in order to perform desired large scale phylogenetic studies using k–NNI local searches.

1.2 Contribution of this work

We provide efficient algorithms for local search heuristics based on 1, 2 and 3–NNI neighborhoods. In fact, we show that local searches based on 2 and 3–NNI neighborhoods are asymptotically just as efficient as those based on 1–NNI, even though they search a much larger neighborhood of trees. For convenience assume that the size of the r given gene trees differs by a constant factor from the size of the resulting species tree, which we denote by n. Local searches based on 1, 2 and 3–NNI respectively induce neighborhoods of size $\Theta(n)$, $\Theta(n^2)$ and $\Theta(n^3)$; and hence, best known (brute-force) solutions for the 1, 2, and 3–NNI local search problems require $O(rn^2)$, $O(rn^3)$, and $O(rn^4)$ time respectively. We provide algorithms that solve the local search problems for both 2 and 3 NNI-neighborhoods in $O(rn^2)$ time.

Furthermore, we show that each subsequent 1, 2, or 3–NNI local search can be solved in $O(rn)$ time. In summary, for all three neighborhoods, the total complexity of a heuristic search involving p local search steps is $O(rn(n + p))$. Thus, if $p \geq n$,

which largely holds true in practice, then the amortized time complexity per local search step is linear in the input size. Consequently, our algorithms provide a total speed-up of $\Theta(\min\{n,p\})$, $\Theta(n \times \min\{n,p\})$, and $\Theta(n^2 \times \min\{n,p\})$ for heuristics that are based on 1, 2 and 3–NNI local searches respectively. It is interesting to note that for 2 and 3–NNI, the complexity of our algorithms is in fact sub-linear in the size of the corresponding neighborhoods. The substantially enlarged neighborhoods, and the exceptional speed-up achieved make the gene-duplication problem more tractable for large-scale phylogenetic analyses.

2 Basic Notation and Preliminaries

In this section we first introduce basic definitions and notation, and then the necessary preliminaries required for this work.

2.1 Basic Definitions and Notation

A *tree* T is a connected graph with no cycles, consisting of a node set $V(T)$ and an edge set $E(T)$. T is *rooted* if it has exactly one distinguished node called the *root* which we denote by $\mathsf{Ro}(T)$. Let T be a rooted tree. We define $\leq_T$ to be the partial order on $V(T)$ where $x \leq_T y$ if y is a node on the path between $\mathsf{Ro}(T)$ and x. The set of minima under $\leq_T$ is denoted by $\mathsf{Le}(T)$ and its elements are called *leaves*. If $\{x,y\} \in E(T)$ and $x \leq_T y$ then we call y the *parent* of x denoted by $\mathsf{Pa}_T(x)$ and we call x a *child* of y. The set of all children of y is denoted by $\mathsf{Ch}_T(y)$. If two nodes in T have the same parent, they are called *siblings*. The *least common ancestor* of a non-empty subset $L \subseteq \mathsf{V}(T)$, denoted as $\mathrm{lca}(L)$, is the unique smallest upper bound of L under $\leq_T$. A *subtree* of T rooted at node $y \in V(T)$, denoted by T_y, is the tree induced by $\{x \in V(T) : x \leq y\}$. T is fully *binary* if every node has either zero or two children. Throughout this paper, the term tree refers to a rooted fully binary tree.

2.2 The Gene Duplication Problem

We now introduce necessary definitions to state the gene-duplication problem. A *species tree* is a tree that depicts the evolutionary relationships of a set of species. Given a gene family for a set of species, a *gene tree* is a tree that depicts the evolutionary relationships among the sequences encoding only that gene family in the given species. Thus, the nodes in a gene tree represent genes. In order to compare a gene tree G with a species tree S a mapping from each gene $g \in V(G)$ to the most recent species in S that could have contained g is required.

Definition 1 (Mapping). *A* leaf-mapping $\mathcal{L}_{G,S}\colon \mathsf{Le}(G) \to \mathsf{Le}(S)$ *specifies, for each gene g the species from which it was sampled. The extension $\mathcal{M}_{G,S}\colon V(G) \to V(S)$ of $\mathcal{L}_{G,S}$ is the* mapping *defined by $\mathcal{M}_{G,S}(g) = \mathrm{lca}(\mathcal{L}_{G,S}(\mathsf{Le}(G_g))$.*

Note: For any node $s \in V(S)$, $\mathcal{M}^{-1}_{G,S}(s)$ denotes the set of nodes in G that map to node $s \in V(S)$ under the mapping $\mathcal{M}_{G,S}$.

Definition 2 (Comparability). *The trees G and S are* comparable *if there exists a leaf-mapping $\mathcal{L}_{G,S}$. A set of gene trees $\mathcal{G}$ and S are* comparable *if each gene tree in $\mathcal{G}$ is comparable with S.*

Throughout this paper we use the following terminology: $\mathcal{G}$ is a set of gene trees that is comparable with a species tree S, and $G \in \mathcal{G}$.

Definition 3 (Duplication). *A node $v \in V(G)$ is a* (gene) duplication *if $\mathcal{M}_{G,S}(v) \in \mathcal{M}_{G,S}(\mathsf{Ch}(v))$ and we define $\mathsf{Dup}(G,S) = \{g \in V(G) \colon$ g is a duplication $\}$.*

Definition 4 (Reconciliation cost). *We define reconciliation costs for gene and species trees as follows:*

1. *$\Delta(G,S) = |\,\mathsf{Dup}(G,S)|$ is the* reconciliation cost from G to S.
2. *$\Delta(\mathcal{G},S) = \sum_{G\in\mathcal{G}} \Delta(G,S)$ is the* reconciliation cost *from $\mathcal{G}$ to S.*
3. *Let $\mathcal{T}$ be the set of species trees that is comparable with $\mathcal{G}$. We define $\Delta(\mathcal{G}) = \min_{S\in\mathcal{T}} \Delta(\mathcal{G},S)$ to be the* reconciliation cost of $\mathcal{G}$.

Problem 1 (Duplication)
Instance: *A set $\mathcal{G}$ of gene trees.*
Find: *A species tree S^* comparable with $\mathcal{G}$, such that $\Delta(\mathcal{G},S^*) = \Delta(\mathcal{G})$.*

2.3 Local Search Problems

Here we first provide the definition of an NNI edit operation [1, 6] and then formulate the related local search problems that were motivated in the Introduction.

Definition 5 (NNI operation). *Let T be a tree. For technical reasons we first define the set valid$(T) = V(T) \setminus \{\{\mathsf{Ro}(T)\} \cup \mathsf{Ch}(\mathsf{Ro}(T))\}$ and call its elements* valid nodes in T. *Now, for $y \in$ valid(T) we denote by $\mathsf{NNI}_T(y)$ the tree that is obtained from T by swapping the subtrees T_x and T_y where x is the sibling of $\mathsf{Pa}(y)$. We say that the tree $\mathsf{NNI}_T(y)$ is obtained from T by a* nearest neighbor interchange (NNI) operation on y *(an example is depicted in Fig. 1).*

In the remainder of this paper, whenever we write $\mathsf{NNI}_T(y)$ we assume that $y \in$ *valid*(T).

Definition 6 (k–NNI neighborhood). *The k–*NNI neighborhood *of a tree T is defined to be the set of all trees that can be obtained by performing at most k successive* NNI *operations on T. The k–*NNI *neighborhood of T is denoted by $k-\mathsf{NNI}_T$.*

Thus, for instance, $1-\mathsf{NNI}_T$ (or simply NNI_T) is the set $\{\mathsf{NNI}_T(y) \colon y \in \mathit{valid}(T)\}$.

Problem 2 (k–NNI–Search)
Instance: *A set $\mathcal{G}$ of gene trees, and a comparable species tree S.*
Find: *A tree $T^* \in k-\mathsf{NNI}_S$ such that $\Delta(\mathcal{G},T^*) = \min_{T\in k-\mathsf{NNI}_S} \Delta(\mathcal{G},T)$.*

In the next section we study the structural properties of 1, 2 and 3–NNI–Search problems. In Section 4 we develop our algorithm for 2–NNI–Search. Our algorithm for further speed-up of the p step 1 and 2–NNI heuristic search appears in Section 5. A description of our algorithm for the 3–NNI–Search problem, and its further speed-up appears in Section 6. Concluding remarks appear in Section 7.

3 Structural Properties

In the following we study the effects of an NNI operation on the mapping $\mathcal{M}_{G,S}$ and on the gene duplication status of nodes from G. Given G and S, consider an NNI operation that changes tree S into tree $S' = \mathrm{NNI}_S(y)$. Figure 1 depicts this situation. Figure 1 also depicts the naming convention that we follow for nodes in S before and after an NNI operation. Essentially, our naming convention preserves the name of each species tree node.
Note: In the interest of brevity, all lemmas in this paper appear with proofs omitted.

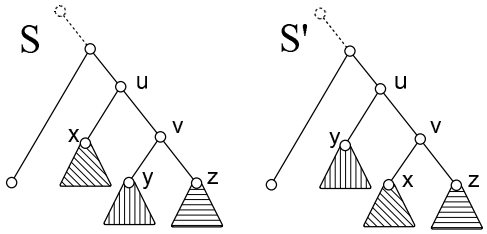

Fig. 1. The tree $S' = \mathrm{NNI}_S(y)$ is obtained by swapping the subtrees S_x and S_y.

Lemma 1. *$\mathcal{M}^{-1}_{G,S}(s) = \mathcal{M}^{-1}_{G,S'}(s)$, for each $s \in V(S) \setminus \{u, v\}$ (see Figure 1).*

Definition 7. *For each $s \in valid(S)$ we define $diff_S(s) = \Delta(\mathcal{G}, S) - \Delta(\mathcal{G}, \mathrm{NNI}_S(s))$.*

Lemma 2. *Let $s \in valid(S)$, p and t be the siblings of s and $\mathrm{Pa}_S(s)$ in S, and p' and t' be the siblings of s and $\mathrm{Pa}_{S'}(s)$ in S' respectively. If $s \in valid(S')$, $\mathrm{Le}(S_t) = \mathrm{Le}(S'_{t'})$, $\mathrm{Le}(S_s) = \mathrm{Le}(S'_s)$, and $\mathrm{Le}(S_p) = \mathrm{Le}(S'_{p'})$, then $diff_S(s) = diff_{S'}(s)$.*

Definition 8. *Given $s \in valid(S)$, let a and b be the siblings of $\mathrm{Pa}_S(s)$ and s respectively. We define $ind_S(s) = valid(S) \setminus (\{a, b, s, \mathrm{Pa}_S(s)\} \cup \mathrm{Ch}_S(s) \cup \mathrm{Ch}_S(a) \cup \mathrm{Ch}_S(b))$, and say that the nodes in $ind_S(s)$ are independent with respect to node s in S.*

Essentially, the nodes in $ind_S(s)$ are important because they satisfy the property in Lemma 3. In the remainder of this paper, whenever we write $ind_S(s)$ we assume that $s \in valid(S)$. A key idea in our algorithms is that when an NNI operation is performed, much of the information computed for the original tree remains the same even for the new tree. This idea is formally captured in Lemma 3. It can be derived based on Lemma 2.

Lemma 3. *If $s \in valid(S') \cap ind_S(y)$, then $diff_{S'}(s) = diff_S(s)$.*

The next two lemmas follow more or less from the definition of $ind_S(s)$, and they are crucial for Lemma 6.

Lemma 4. *$|valid(S) \setminus ind_S(s)| \leq 10$.*

Lemma 5. *If $s \in valid(S)$, then $|\{t \in valid(S) \colon s \notin ind_S(t)\}| \leq 10$.*

4 Solving the 2–NNI–Search Problem

In this section we describe our algorithm to solve the 2–NNI–Search problem. The first step in our algorithm is to compute the value $\mathit{diff}_S(s)$ for each $s \in \mathit{valid}(S)$. This already gives a solution to the 1–NNI–Search problem. Subsequently, the algorithm computes a lowest reconciliation cost tree in $2-\mathrm{NNI}_S \setminus \mathrm{NNI}_S$. All trees in $2-\mathrm{NNI}_S \setminus \mathrm{NNI}_S$, are obtained by performing exactly 2 successive NNI operations on tree S. Consider some tree $T \in 2-\mathrm{NNI}_S \setminus \mathrm{NNI}_S$. Then there must exist two nodes $s, t \in V(S)$ such that $T = \mathrm{NNI}_{T'}(t)$, and $T' = \mathrm{NNI}_S(s)$. Now there are two possible cases:
(i) $t \in \mathit{ind}_S(s)$, or (ii) $t \notin \mathit{ind}_S(s)$.

The overall idea of our algorithm is as follows: We compute a minimum reconciliation cost tree among the trees that satisfy Case (i) above, and a minimum reconciliation cost tree among the trees that satisfy Case (ii). We also compute a minimum reconciliation cost tree in NNI_S. The best tree among these three trees must be a minimum reconciliation cost tree in $2-\mathrm{NNI}_S$. The lemmas that follow, allow us to efficiently compute a minimum reconciliation cost tree in $2-\mathrm{NNI}_S \setminus \mathrm{NNI}_S$.

Lemma 6. *Let A denote the set of the first 11 nodes valid in S arranged according to decreasing values of $\mathit{diff}_S(s)$. Let $\Gamma = \{T = \mathrm{NNI}_{T'}(t) \colon T' = \mathrm{NNI}_S(s)$, and $t \in \mathit{ind}_S(s)\}$. Let $R^* \in \Gamma$ with minimum reconciliation cost. Then, there exists a pair of nodes $a, b \in A$ such that $b \in \mathit{ind}_S(a)$, $R = \mathrm{NNI}_{R'}(b)$, $R' = \mathrm{NNI}_S(a)$, and $\Delta(\mathcal{G}, R^*) = \Delta(\mathcal{G}, R)$.*

Lemma 7. *Let $t \in \mathit{valid}(T)$ where $T = \mathrm{NNI}_S(s)$, such that $t \notin \mathit{ind}_S(s)$. Let a be the sibling of $\mathrm{Pa}_S(s)$, and b be the sibling of s in S. Then, $t \in \{\{a, b, s, \mathrm{Pa}_S(s)\} \cup \mathrm{Ch}_S(s) \cup \mathrm{Ch}_S(a) \cup \mathrm{Ch}_S(b)\}$.*

We can now present our algorithm to solve the 2–NNI–Search problem. We call this algorithm ALG-2-NNI , and a description of this algorithm appears as Algorithm 1.

The input for Algorithm 1 is a set of gene trees $\mathcal{G}$, and a species tree S. Let $n = |\mathrm{Le}(S)|$, and $r = |\mathcal{G}|$. To simplify the complexity analysis, we shall assume that all input gene trees have almost the same size. Thus, let $m = |\mathrm{Le}(S)| + |\mathrm{Le}(G)|$ for some $G \in \mathcal{G}$. Note: the speed-up obtained by our algorithm does not depend on this simplifying assumption.

Theorem 1. *Algorithm 1 solves the 2–NNI–Search problem in $O(rmn)$ time.*

Proof. (Correctness) Each tree $T \in 2-\mathrm{NNI}_S$ belongs to one of the following cases:

1. $T \in \mathrm{NNI}_S$: The tree T_1 computed in Algorithm 1 is a tree with minimum reconciliation cost among all trees in NNI_S.
2. $T \in 2-\mathrm{NNI}_S \setminus \mathrm{NNI}_S$: There exist two nodes $s, t \in V(S)$ such that $T = \mathrm{NNI}_{T'}(t)$, and $T' = \mathrm{NNI}_S(s)$. We now have two possible cases:
   (a) $t \in \mathit{ind}_S(s)$: According to Lemma 6, the tree T_2 computed by Algorithm 1 must be a minimum reconciliation cost tree among all trees in this case.
   (b) $t \notin \mathit{ind}_S(s)$: According to Lemma 7, the tree T_3 computed by Algorithm 1 must be a minimum reconciliation cost tree among all trees in this case.

Algorithm 1 ALG-2-NNI

Input: A set of gene trees $\mathcal{G}$, and, a species tree S
Output: A tree $T^* \in k - \text{NNI}_S$ such that $\Delta(\mathcal{G}, T^*) = \min_{T \in k-\text{NNI}_S} \Delta(\mathcal{G}, T)$

for each $s \in \mathit{valid}(S)$ **do**
  Compute the value $\mathit{diff}_S(s)$.
Let $\alpha \in \arg\max_{a \in \mathit{valid}(S)} \mathit{diff}_S(a)$, and set $T_1 = \text{NNI}_S(\alpha)$.
Let A denote the set of the first 11 nodes valid in S arranged according to decreasing values of $\mathit{diff}_S(s)$.
$(\alpha, \beta) \in \arg\max_{(a,b):\ a,b \in A,\ b \in \mathit{ind}_S(a)} \mathit{diff}_S(a) + \mathit{diff}_S(b)$.
Set $T = \text{NNI}_S(\alpha)$ and $T_2 = \text{NNI}_T(\beta)$.
Set $T_3 = T_2$.
for each $s \in \mathit{valid}(S)$ **do**
  Let a be the sibling of $\text{Pa}_S(s)$, and b be the sibling of s in S. Set $T = \text{NNI}_S(s)$.
  for $t \in \mathit{valid}(T) \cap \{\{a, b, s, \text{Pa}_S(s)\} \cup \text{Ch}_S(s) \cup \text{Ch}_S(a) \cup \text{Ch}_S(b)\}$ **do**
    $R = \text{NNI}_T(t)$.
    if $\Delta(\mathcal{G}, T_3) > \Delta(\mathcal{G}, T)$ **then**
      Set $T_3 = T$.
return an element of $\arg\min_{T \in \{T_1, T_2, T_3\}} \Delta(\mathcal{G}, T)$.

Therefore, a minimum reconciliation cost tree among T_1, T_2, T_3 must be a solution to the 2–NNI–Search problem.

(Complexity) Computing the tree T_1 involves computing the $\mathit{diff}_S(s)$ value for each $s \in \mathit{valid}(S)$, and identifying the node a for which $\mathit{diff}_S(a)$ is maximum. Computing the reconciliation cost for a given species tree takes $O(rm)$ time. Therefore, computing T_1 takes $O(rmn)$ time.

After T_1 has been computed, computing the tree T_2 involves creating the set A (which takes $O(n)$ time), and then evaluating every possible 2-element ordered pair from A. Each evaluation takes $O(1)$ time, and the number of possible ordered pairs is $O(|A|^2)$ i.e. $O(1)$. Therefore, computing T_2 (after having computed T_1) requires $O(n)$ time.

Computing T_3 involves evaluating the reconciliation costs of at most $10 \times n$ i.e. $O(n)$ trees, and then picking the best tree among these. Therefore, computing T_3 requires $O(rmn)$ time.

In conclusion, the time complexity of Algorithm 1 is $O(rmn)$. □

5 Further Speed-up for 1 and 2–NNI Heuristics

As mentioned earlier, standard local search heuristics for the Duplication problem, involve solving many instances of these local search problems. Consider a heuristic search involving p instances of the local search problem, then, using our faster algorithm for the 2–NNI–Search problem allows both 1 and 2–NNI based heuristics to run in $\Theta(prmn)$ time. We will now show that the 1, 2–NNI based heuristics can, in fact, both be executed in $O(rm(n + p))$ time.

5.1 Heuristics based on 1–NNI

Existing algorithms for the 1–NNI–Search (or simply NNI–Search) problem have a time complexity of $O(rmn)$, and hence they solve the NNI based heuristic problem in $O(rpmn)$ time. Our algorithm to solve the NNI–Search problem involves computing the the value $\mathit{diff}_S(s)$ for each $s \in \mathit{valid}(S)$, and then picking a tree T such that $T = \mathrm{NNI}_S(\alpha)$ where $\alpha = \arg\max_{a \in \mathit{valid}(S)} \mathit{diff}_S(a)$. This also requires $O(rmn)$ time. However, this approach allows us to reuse most of the previously computed information in subsequent iterations of the local search.

Let T denote a minimum reconciliation cost tree in NNI_S. Then, there exists a node a such that $T = \mathrm{NNI}_S(a)$. For the next iteration we must compute a minimum reconciliation cost tree in NNI_T. As seen earlier, this involves computing the value $\mathit{diff}_T(s)$ for each $s \in \mathit{valid}(T)$. Let $\Gamma = \mathit{valid}(T) \cap \mathit{ind}_S(a)$. Then, by Lemma 3 we know that $\mathit{diff}_T(s) = \mathit{diff}_S(s)$, for all $s \in \Gamma$. Therefore, for all $s \in \Gamma$ we can reuse the values from the previous iteration. In other words we must only compute the value $\mathit{diff}_T(s)$ for all $s \in \mathit{valid}(T) \setminus \Gamma$. It follows directly from Lemma 7 that if $\Phi = \mathit{valid}(T) \setminus \Gamma$, then $|\Phi| \leq 10$.

This means that for each subsequent iteration of the NNI local search, we must compute the reconciliation costs for at most 10 trees. Thus, once the first NNI local search problem has been solved in $O(rmn)$ time, each subsequent local search instance can be solved in $O(rm)$ time. This gives a total time complexity of $O(rm(n+p))$, which gives a speed-up by a factor of $\Theta(\min\{n,p\})$ over existing solutions.

5.2 Heuristics based on 2–NNI

Let T denote a minimum reconciliation cost tree in $2-\mathrm{NNI}_S$. For the next iteration of this local search, we wish to find a tree U with minimum reconciliation cost in $2-\mathrm{NNI}_T$. According to our algorithm (see Algorithm 1) computing U involves computing the trees $T_1, T_2, T_3 \in 2-\mathrm{NNI}_T$. We now show how to compute each of these three special trees in $O(rm)$ time by reusing previously computed information.

There exist two nodes a, b such that $T' = \mathrm{NNI}_S(a)$ and $T = \mathrm{NNI}_{T'}(b)$. Computing the tree T_1 involves computing the value $\mathit{diff}_T(s)$ for all nodes $s \in \mathit{valid}(T)$. Since a and b are known (from the previous iteration of the local search), the method used for 1–NNI above can be used to obtain the values $\mathit{diff}_{T'}(s)$ for all $s \in \mathit{valid}(T')$ in $O(rm)$ time. Once this is done, the same algorithm is reapplied to compute the values $\mathit{diff}_T(s)$ for all $s \in \mathit{valid}(T)$. This step also takes $O(rm)$ time. Hence, the tree T_1 can be computed in $O(rm) + O(rm)$ i.e. $O(rm)$ time.

Once all the $\mathit{diff}_T(s)$ values have been obtained for all $s \in \mathit{valid}(T)$, computing the tree T_2 takes $O(n)$ time (see the complexity analysis in the proof of Theorem 1).

In order to compute the tree T_3, we first compute the tree $\mathrm{NNI}_T(s)$ for each $s \in \mathit{valid}(T)$ and then compute the scores for at most 10 trees derived from $\mathrm{NNI}_T(s)$, for each $s \in \mathit{valid}(T)$ (see Algorithm 1). We will show how to efficiently obtain all these $O(10n)$ scores by reusing the scores computed in the previous iteration of the local search. It is sufficient to show how to obtain these scores for the tree $T' = \mathrm{NNI}_S(a)$, because the exact same procedure can be applied again on T' to obtain the scores for the tree $T = \mathrm{NNI}_{T'}(b)$.

Let c, d be two nodes such that $R' = \mathsf{NNI}_{T'}(c)$ and $R = \mathsf{NNI}_{R'}(d)$. Since we wish to compute the tree corresponding to T_3, we may assume that $d \notin \mathit{ind}_{R'}(c)$. There are three possible cases:

1. $c, d \in \mathit{ind}_S(a)$: Let $Q = \mathsf{NNI}_S(c)$. In this case we have the values $\mathit{diff}_S(c)$ and $\mathit{diff}_Q(d)$ computed from the previous iteration. Since $c \in \mathit{ind}_S(a)$, by Lemma 3 we have $\mathit{diff}_{T'}(c) = \mathit{diff}_S(c)$. It can be shown that there are $O(1)$ candidates for d such that $\mathit{diff}_{R'}(d) \neq \mathit{diff}_Q(d)$. Thus, in case $\mathit{diff}_{R'}(d) \neq \mathit{diff}_Q(d)$, by Lemma 5 there are only $O(1)$ candidates for c as well, and hence the score for each such pair can be computed in $O(rm)$ time. Otherwise, the previously computed scores can be reused, which takes $O(n)$ time. This gives a total time complexity of $O(rm)$.
2. $c \in \mathit{ind}_S(a)$, $d \notin \mathit{ind}_S(a)$: d may be either valid or invalid in S. If $d \notin \mathit{valid}(S)$, then there are no more than two candidates for d (since $d \in \mathit{valid}(R')$, and R' is obtained from S by no more than two NNI operations). Otherwise, there are $O(1)$ candidates each for d (see Lemma 4). Since $d \notin \mathit{ind}_{R'}(c)$, Lemma 5 implies that there are $O(1)$ candidates for c. Hence, we only need to compute $O(1)$ scores.
3. $c \notin \mathit{ind}_S(a)$: c may be either valid or invalid in S. If $c \notin \mathit{valid}(S)$, then there is exactly one candidate for c (since $c \in \mathit{valid}(T')$). Otherwise, there are at most 10 candidates each for c and d (see Lemma 4). Hence, we only need to compute $O(1)$ scores.

Thus, T_3 can be computed in $O(rm)$ time as well, which in turn implies that a minimum reconciliation cost tree in $2 - \mathsf{NNI}_T$ can be computed in $O(rm)$ time. This gives a total time complexity of $O(rm(n+p))$ for 2–NNI based heuristics, which gives a speed-up by a factor of $\Theta(n \times \min\{n, p\})$ over the naive solution.

6 Optimizing the 3–NNI–Search Problem

The main idea behind our algorithms for the 1 and 2–NNI–Search problems, as well as their speed-up, is that when an NNI operation is performed on a tree, it only affects the mapping in a small, constant sized region of the tree. Since the reconciliation cost depends only on the mapping from the gene trees, in the new species tree thus obtained, much of the information computed for the original tree remains valid. This idea applies equally well for solving the k–NNI–Search problem, for $k > 2$, but the algorithm becomes progressively more convoluted as k increases. However, for the special case of $k = 3$, the algorithm for 2–NNI–Search extends in a rather straightforward manner.

The trees in $3 - \mathsf{NNI}_S$ must be in at least one of $2 - \mathsf{NNI}_S$, or $3 - \mathsf{NNI}_S \setminus 2 - \mathsf{NNI}_S$. We have already seen how to obtain a minimum reconciliation cost tree in $2 - \mathsf{NNI}_S$. Therefore, the problem is to find a minimum reconciliation cost tree in $3 - \mathsf{NNI}_S \setminus 2 - \mathsf{NNI}_S$. All the trees in $3 - \mathsf{NNI}_S \setminus 2 - \mathsf{NNI}_S$, are obtained by performing exactly 3 successive NNI operations on tree S. Consider some tree $T \in 3 - \mathsf{NNI}_S \setminus 2 - \mathsf{NNI}_S$. Then there must exist three nodes $s, t, u \in V(S)$ such that $T = \mathsf{NNI}_{T'}(u)$, $T' = \mathsf{NNI}_{T''}(t)$, and $T'' = \mathsf{NNI}_S(s)$. We now have six cases, exactly one of which must be true.

1. $t \in \mathit{ind}_S(s), u \in \mathit{ind}_{T''}(t) \cap \mathit{ind}_S(s)$
2. $t \in \mathit{ind}_S(s), u \in \mathit{ind}_S(s) \setminus \mathit{ind}_{T''}(t)$

3. $t \in \mathit{ind}_S(s), u \in \mathit{ind}_{T''}(t) \setminus \mathit{ind}_S(s)$
4. $t \in \mathit{ind}_S(s), u \notin \mathit{ind}_{T''}(t) \cup \mathit{ind}_S(s)$
5. $t \notin \mathit{ind}_S(s), u \in \mathit{ind}_{T''}(t) \cap \mathit{ind}_S(s)$
6. $t \notin \mathit{ind}_S(s), u \notin \mathit{ind}_{T''}(t) \cap \mathit{ind}_S(s)$

If we can calculate a minimum reconciliation cost tree separately for each of these six cases, then the tree with minimum cost among these six trees will be a minimum reconciliation cost tree in $3 - \mathsf{NNI}_S \setminus 2 - \mathsf{NNI}_S$.

It can be shown that a minimum reconciliation cost tree can be obtained for each of the six cases in $O(rmn)$ time (details omitted for brevity). This gives us an $O(rmn)$ time algorithm for the 3–NNI–Search problem.

The algorithm used to obtain further speed-up for 2–NNI based heuristics also extends in a similar fashion to 3–NNI based heuristics. This gives a total time complexity of $O(rm(n+p))$ for the 3–NNI based heuristic.

7 Outlook and Conclusion

We introduced algorithms that significantly speed up NNI based local search heuristics for the duplication problem. These algorithms extend naturally to local search problems based on the *Edge Contraction and Refinement (ECR)* edit operation [11, 12]. Thus, heuristic searches involving p instances of the 1, 2, or 3–ECR–Search problems can all be completed in $O(rm(n+p))$ time as well.

Our algorithms form the basis for extremely efficient local search heuristics. In particular, our 2 and 3–NNI local search algorithms can greatly improve on the performance of classical 1–NNI local search heuristics, without sacrificing efficiency. The real power of our algorithms can be best exploited as part of a heuristic that mixes 1, 2, and 3–NNI local searches with SPR and TBR local searches (see [22]). Such a heuristic would be both fast and effective, which would enable much larger analyses to be performed within a reasonable time. In future work, these techniques might set base for algorithmic theory that identifies a much broader class of local search problems which can be solved more efficiently.

Acknowledgements. This work was supported in part by NSF grant no. 0334832. The authors also wish to thank the anonymous reviewers for their invaluable comments.

References

1. B. L. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5:1–13, 2001.
2. M. S. Bansal, J. G. Burleigh, O. Eulenstein, and A. Wehe. Heuristics for the gene-duplication problem: A $\Theta(n)$ speed-up for the local search. In *RECOMB*, pages 238–252, 2007.
3. M. S. Bansal and O. Eulenstein. An $\Omega(n^2/\log n)$ speed-up of TBR heuristics for the gene-duplication problem. In *WABI*, pages 124–135, 2007.
4. M. A. Bender and M. Farach-Colton. The LCA problem revisited. In *LATIN*, pages 88–94, 2000.
5. P. Bonizzoni, G. D. Vedova, and R. Dondi. Reconciling a gene tree to a species tree under the duplication cost model. *Theor. Comput. Sci.*, 347(1-2):36–53, 2005.

6. M. Bordewich and C. Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Combinatorics*, 8:409–423, 2004.
7. K. Chen, D. Durand, and M. Farach-Colton. Notung: a program for dating gene duplications and optimizing gene family trees. *Journal of Computational Biology*, 7:429–447, 2000.
8. J. A. Cotton and R. D. M. Page. *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, chapter Tangled tales from multiple markers: reconciling conflict between phylogenies to build molecular supertrees, pages 107–125. Springer-Verlag, 2004.
9. B. DasGupta, X. He, T. Jiang, M. Li, J. Tromp, and L. Zhang. On distances between phylogenetic trees. In *SODA*, pages 427–436, 1997.
10. M. Fellows, M. Hallett, C. Korostensky, and U. Stege. Analogs & duals of the mast problem for sequences & trees. In *European Symposium on Algorithms (ESA)*, pages 103–114, 1998.
11. G. Ganapathy, V. Ramachandran, and T. Warnow. Better hill-climbing searches for parsimony. In *WABI*, pages 245–258, 2003.
12. G. Ganapathy, V. Ramachandran, and T. Warnow. On contract-and-refine transformations between phylogenetic trees. In *SODA*, pages 900–909, 2004.
13. M. Goodman, J. Czelusniak, G. W. Moore, A. E. Romero-Herrera, and G. Matsuda. Fitting the gene lineage into its species lineage. a parsimony strategy illustrated by cladograms constructed from globin sequences. *Systematic Zoology*, 28:132–163, 1979.
14. P. Górecki and J. Tiuryn. On the structure of reconciliations. In *RECOMB Comparative Genomics Workshop*, 2004.
15. R. Guigó, I. Muchnik, and T. F. Smith. Reconstruction of ancient molecular phylogeny. *Molecular Phylogenetics and Evolution*, 6(2):189–213, 1996.
16. M. T. Hallett and J. Lagergren. New algorithms for the duplication-loss model. In *RECOMB*, pages 138–146, 2000.
17. B. Ma, M. Li, and L. Zhang. From gene trees to species trees. *SIAM J. Comput.*, 30(3):729–752, 2000.
18. B. Mirkin, I. Muchnik, and T. F. Smith. A biology consistent model for comparing molecular phylogenies. *Journal of Computational Biology*, 2(4):493–507, 1995.
19. R. D. M. Page. Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas. *Systematic Biology*, 43(1):58–77, 1994.
20. R. D. M. Page. GeneTree: comparing gene and species phylogenies using reconciled trees. *Bioinformatics*, 14(9):819–820, 1998.
21. R. D. M. Page. Extracting species trees from complex gene trees: reconciled trees and vertebrate phylogeny. *Molecular Phylogenetics and Evolution*, 14:89–106, 2000.
22. R. D. M. Page and M. A. Charleston. From gene to organismal phylogeny: reconciled trees and the gene tree/species tree problem. *Molec. Phyl. and Evol.*, 7:231–240, 1997.
23. R. D. M. Page and J. Cotton. Vertebrate phylogenomics: reconciled trees and gene duplications. In *Pacific Symposium on Biocomputing*, pages 536–547, 2002.
24. R. D. M. Page and E. C. Holmes. *Molecular evolution: a phylogenetic approach*. Blackwell Science, 1998.
25. M. J. Sanderson and M. M. McMahon. Inferring angiosperm phylogeny from EST data with widespread gene duplication. *BMC Evolutionary Biology*, 7 (Suppl 1): S3, 2007.
26. C. Semple and M. Steel. *Phylogenetics*. Oxford University Press, 2003.
27. J. B. Slowinski, A. Knight, and A. P. Rooney. Inferring species trees from gene trees: A phylogenetic analysis of the elapidae (serpentes) based on the amino acid sequences of venom proteins. *Molecular Phylogenetics and Evolution*, 8:349–362, 1997.
28. U. Stege. Gene trees and species trees: The gene-duplication problem in fixed-parameter tractable. In *WADS*, pages 288–293, 1999.
29. L. Zhang. On a Mirkin-Muchnik-Smith conjecture for comparing molecular phylogenies. *Journal of Computational Biology*, 4(2):177–187, 1997.