Analogs & Duals of the MAST Problem for Sequences & Trees

Michael Fellows[1], Michael Hallett[2], Chantal Korostensky[2], Ulrike Stege[2]

[1] Department of Computer Science, University of Victoria
Victoria, B.C. Canada V8W 3P6, `mfellows@csr.uvic.ca`
[2] Computational Biochemistry Research Group, E.T.H. Zürich
Ch-8092 Zürich, Switzerland {`hallett`,`korosten`,`stege`}`@inf.ethz.ch`

Abstract. Two natural kinds of problems about "structured collections of symbols" can be generally refered to as the LARGEST COMMON SUBOBJECT and the SMALLEST COMMON SUPEROBJECT problems, which we consider here as the dual problems of interest. For the case of rooted binary trees where the symbols occur as leaf-labels and a subobject is defined by label-respecting hereditary topological containment, both of these problems are NP-complete, as are the analogous problems for sequences (the well-known LONGEST COMMON SUBSEQUENCE and SHORTEST COMMON SUPERSEQUENCE problems). However, when the trees are restricted by allowing each symbol to occur as a leaf-label at most once (which we call a *phylogenetic tree* or *p-tree*), then the LARGEST COMMON SUBOBJECT problem, better known as the MAXIMUM AGREEMENT SUBTREE (MAST) problem, is solvable in polynomial time. We explore the complexity of the basic subobject and superobject problems for sequences and binary trees when the inputs are restricted to p-trees and p-sequences (*p-sequences* are sequences where each symbol occurs at most once). We prove that the sequence analog of MAST can be solved in polynomial time. The SHORTEST COMMON SUPERSEQUENCE problem restricted to inputs consisting of a collection of p-sequences (pSCS) remains NP-complete, as does the analogous SMALLEST COMMON SUPERTREE problem restricted to p-trees (pSCT). We also show that both problems are hard for the parameterized complexity classes $W[1]$ where the parameter is the number of input trees or sequences. We prove fixed-parameter tractability for pSCS and pSCT when the k input sequences (trees) are restricted to be complete: every symbol of Σ occurs exaxtly once in each object and the question is whether there is a common superobject of size bounded by $|\Sigma| + r$ and the parameter is the pair (k, r). We show that without this restriction, both problems are harder than DIRECTED FEEDBACK VERTEX SET, for which parameterized complexity is famously unresolved. We describe an application of the tractability result for pSCT in the study of gene duplication events, where k and r are naturally small.

1 Introduction

Typically algorithms used for computing phylogenetic relationships are concerned, naturally, with trees where the leaves are labeled from a set Σ of species and each label occurs at most once. However, the study of gene duplication events leads to models that make use of trees where the species labels may occur more than once.

Example 1: Modeling Gene Duplication With Repeat-Labeled Trees. When trying to resolve the species tree for a set of n taxa, one typically creates a set of k gene trees. It is not always the case that the gene trees agree. One such reason is due to paralogous duplications of genes followed by subsequent loss of genes. The papers [GCMRM79,GMS96,Z97] provide a formal model for evaluating the "cost" of rectifying the k gene trees with the correct species (phylogenetic) tree. This model implicitly makes use of trees with repeated leaf labels.

For problems about sequences, we usually assume that the sequences of interest will contain occurences of the same symbol many times. But there are some applications where attention may be restricted to sequences x where any symbol occuring in x occurs at most once.

Example 2: Scheduling Manufacturing Operations. The SHORTEST COMMON SUPERSEQUENCE problem can be applied to scheduling. Each symbol of the alphabet Σ corresponds to an operation in the sequential process of manufacturing an object. The input to the problem corresponds to the manufacture sequences for a number of different objects that share the same production line. A common supersequence then corresponds to a schedule of operations for the production line that allows all of the different objects to be manufactured. It may reasonably be the case that each object to be manufactured requires any given operation to be applied at most once.

Restricting attention to trees or sequences without repeated symbols seems to have a significant effect on problem complexity. In order to discuss the issue we introduce the following terminology. All trees in this paper are binary unless stated otherwise. A *p-tree* is a rooted tree where the leaves are labeled from an alphabet Σ, and where no symbol in Σ is used more than once as a label. An *rl-tree* is a rooted tree with leaves labeled from Σ, where labels may be repeated. Unless stated otherwise n is used to denote the size of Σ, $|\Sigma| = n$. Say that a string of symbols (or sequence) $x \in \Sigma^*$ is a *p-string* (*p-sequence*) if no symbol of Σ occurs more than once in x. We call x a *complete* p-sequence if each symbol of the alphabet occurs exactly once in x. To emphasize the analogy with trees, say that x is an *rl-string* if it is a string in the usual sense, where symbols of Σ may be repeated. If x and y are strings, then we write $x \subseteq y$ to denote that x is a subsequence of y. If X and Y are rl-trees, then $X \subseteq Y$ denotes that X is contained in Y by topological containment that respects ancestry with label isomorphism at the leaves. For a given tree T, the *size* $|T|$ of T is defined by the number of leaves in T.

The following quartet of fundamental computational problems are natural and important for various applications.

Two Problems About Trees

LARGEST COMMON SUBTREE (LCT)
Input: rl-trees $T_1, ..., T_k$ and a positive integer m
Question: Is there an rl-tree T of size $|T| \geq m$ leaves, with $T \subseteq T_i$ for $i = 1, ..., k$?

Smallest Common Supertree (SCT)
Input: rl-trees $T_1, ..., T_k$ and a positive integer m
Question: Is there an rl-tree T of size $|T| \leq m$ leaves, with $T_i \subseteq T$ for $i = 1, .., k$?

Two Problems About Sequences

Longest Common Subsequence (LCS)
Input: rl-sequences $x_1, ..., x_k$ and a positive integer m
Question: Is there an rl-sequence x, with $|x| \geq m$ and $x \subseteq x_i$ for $i = 1, ..., k$?

Shortest Common Supersequence (SCS)
Input: rl-sequences $x_1, ..., x_k$ and a positive integer m
Question: Is there an rl-sequence x, with $|x| \leq m$ and $x_i \subseteq x$ for $i = 1, ..., k$?

All four of these problems are *NP*-complete (see [M78] for the sequence problems; we prove *NP*-completeness for the tree problems in Section 3). Both LCS and SCS can be solved in $O(n^k)$ time [M78]. We prove that LCT and SCT, the tree analogs of LCS and SCS, have similar polynomial complexity for fixed k (Section 3). The question we explore is

What happens to the complexity of these basic problems when the inputs are restricted to be p-trees and p-sequences?

The Largest Common Subtree problem restricted to p-trees is better known as the Maximum Agreement Subtree (MAST) problem. This is an important problem for which useful polynomial-time algorithms (for trees of bounded degree) have recently been developed. In the study of evolution the situation frequently arises that one has k sets of gene sequence data for n species. A typical approach would be to compute *gene trees* for each of the data sets. For various reasons, these trees will frequently not be in agreement. One may then use an algorithm for MAST to compute a maximum subtree (*species tree*) on which the gene trees agree. Farach et al. have described an algorithm for MAST for k rooted p-trees for n species of maximum degree d that runs in time $O(kn^3 + n^d)$ [FPT95]. A different (and simpler) algorithm with the same time complexity was developed by Bryant [B97]. Przytycka has described a modification of the algorithm of [FPT95] that runs in time $O(kn^3 + k^d)$ [P97].
The above described results on MAST answer our question for one of the four basic problems, but what of the other three?

Main Results. (1) We describe an $O(|\Sigma|^2)$ algorithm to solve the Longest Common Subsequence problem for inputs restricted to p-sequences. (2) We prove that Shortest Common Supersequence and Smallest Common Supertree for inputs restricted to p-sequences and p-trees remain *NP*-complete, and are hard for $W[1]$ when the parameter is the number of input objects. (3) We prove fixed-parameter tractability for Shortest Common Supersequence for p- and rl-sequences when the question is whether there is a common supersequence of length at most $|\Sigma| + r$ parameterised by the number of sequences and r. We present an algorithm with running time $O(k^r \cdot \Sigma)$. (4) We prove fixed-parameter tractability for Shortest Common Supersequence for complete p-sequences and parameter r when the question is whether there is a common supersequence of length $|\Sigma| + r$. (In fact, we prove tractability for a more general, weighted alphabet form of the problem.) (5) We prove fixed-parameter tractability for Shortest Common Supertree for k complete p-trees and parameter is the pair (k, r), when the question is whether there is a common supertree of size $|\Sigma| + r$. Readers not familiar with parameterized complexity are referred to [DF93,DF95a,DF95b,DF95c,DF98].

2 Sequence Analogs of MAST

We consider the following analog of MAST for p-sequences. While not difficult, the positive result shows an interesting parallel between the complexity of sequence problems and the complexity of leaf-labeled tree problems that we will see holds up also for the superobject problems.

Longest Common Subsequence for p-Sequences (pLCS)
Input: p-sequences $x_1, ..., x_k$ over an alphabet Σ, and $m \in Z^+$.
Question: Is there a p-sequence x, $|x| \geq m$, such that $x \subseteq x_i$ for $i = 1, ..., k$?
Theorem. pLCS can be solved in time $O(k \cdot n \log n)$ for an alphabet of size n.
Proof (sketch): The basic structure of the algorithm is based on the partial ordering of the elements of Σ defined: $a \leq b$ if and only if $\forall i$: a precedes b in x_i. (Note that the input being p-sequences is crucial to having this be well-defined.) We simply compute a longest chain in this poset by topological sorting. Including the time to compare elements in the poset, this can be accomplished in time $O(k \cdot n \log n)$ using the appropriate data structures. ∎

3 Tree Analogs of SCS and LCS

The complexity situation for the tree analogs SCT and LCT of the sequence problems SCS and LCS turns out to be almost identical to the situation for these well-studied problems. Both SCT and LCT are *NP*-complete, but can be solved in time $O(n^{k+1})$ for k trees by dynamic programming. A straightforward reduction from LCS encoding sequences as caterpillar trees proves the *NP*-completeness of LCT. An $O(n^{k+1})$-time algorithm for LCT can be given by dynamic programming. Both proofs are omitted.

Theorem 1. *LCT restricted to binary trees is* NP-*complete and, for each fixed k, it can be solved in time $O(n^{k+1})$.*

Theorem 2. *SCT restricted to binary trees is* NP-*complete.*

Proof (sketch): We reduce from the *NP*-complete problem Directed Feedback Vertex Set (DFVS) (see [GJ79]) in two steps. In the first step, we reduce to the SCS problem. In the second step, we transform the SCS instance into an instance of SCT. The DFVS problem takes as input a directed graph $D = (V, A)$ and a positive integer k and asks whether there is a set $V' \subseteq V$ with $|V'| \leq k$ such that $D - V'$ is acyclic. (As a parameterized problem, we take k to be the parameter.)
Let $D = (V, A)$ be a digraph for which we wish to determine if it has a directed

feedback vertex set of size k. The instance of SCS to which we transform this is described as follows. The alphabet is V. Each arc uv of D becomes the length 2 sequence uv. The set of sequences S thus has the same size as the set A of arcs of D. The parameter k' of the image instance is equal to k.
Let x be a solution for the sequence problem, and let V' be the set of vertices (which double as the symbols of the alphabet) that occur more than once in x. Clearly $|V'| \leq k$. We argue that V' covers all the directed cycles in D. If not, then there is a directed cycle C in D involving vertices that occur exactly once in x. Let a be the first vertex of C that occurs in x. But then there is some vertex b of C such that ba is an arc of C and therefore ba is a sequence in S. Since b occurs after a in x this contradicts that x is a solution to the SCS problem.
Conversely, let V' be a k-element directed feedback vertex set for D.
A common supersequence x for the sequences in S can be written $x = x_1 x_2 x_3$ where $x_1 = x_3$ is any permutation of V' and x_2 is a topological sort of $V - V'$. If uv is an arc of D where both u and v belong to V', then clearly uv is a subsequence of x, either by finding u in x_1 and v in x_3 or vice versa. If both u and v belong to $V - V'$ then uv is a subsequence of x_2. The two remaining cases (where exactly one of u and v belongs to V') are equally easy to check.

The second step of theorem transforms an instance of SCS to SCT. An instance of SCS consists of a set $\mathcal{X}$ of sequences over an alphabet Σ and a positive integer l. A sequence $x_i \in \mathcal{X}$ of length t, where x_i consists of the sequence of symbols of $x_i = x_i[1] \cdots x_i[t]$ is transformed to a tree T_i represented by the parenthesized expression $(((\cdots((F, x_i[1]), x_1[2]) \cdots) x_i[t]), \$)$ where F is a complete binary tree having more than l leaves labeled from a set of new symbols that we will also denote F, and where \$ is another new symbol. The alphabet of leaf labels of the trees is thus $\Sigma' = \Sigma \cup F \cup \{\$\}$. The theorem follows from Lemma 1, for which the proof is omitted. ∎

Lemma 1. *If $\mathcal{X}$ is a set of p-sequences $\mathcal{X} = \{x_i : 1 \leq i \leq m\}$ over an alphabet Σ, $|\Sigma| = n$, where each symbol of Σ occurs in at least one sequence, r is a positive integer, and $\mathcal{T} = \{T_i : 1 \leq i \leq m\}$ is the set of p-trees that are the images of the transformation from* SCS *to* SCT*, Σ' is the transformed alphabet, $|\Sigma'| = n'$, then there is a common supersequence x of the sequences in $\mathcal{X}$ of length at most $n + r$ if and only if there is an rl-tree T having size at most $n' + r$ that is a common supertree for the trees in $\mathcal{T}$.*

We remark that part 1 above gives a much simpler proof of the *NP*-completeness for SCS than [M78].

Theorem 3. *For fixed k, SCT on binary trees can be solved in time $O(n^{k+1})$.*

Proof (sketch): Using dynamic programming we can calculate, for each coordinate k-tuple of vertices $c = (u_1, ..., u_k)$, where u_i is a vertex of T_i for $i = 1, ..., k$, the number of leaves $l(c)$ (and a representative, for the search algorithm) of a smallest common binary supertree for the subtrees rooted at the vertices u_i. A solution for the given input can be found in the resulting table at the coordinate k-tuple of roots of the T_i.

The case of $k = 2$ is instructive and generalizes easily to larger values of k. Suppose the vertices u_i of T_i, for $i = 1, 2$ have children u_i^L and u_i^R. The basis for the dynamic programming recurrence is the observation that if T is a common supertree of T_1 and T_2, then there are essentially 7 different ways that the T_i can be embedded in a binary supertree T: (1) with T_1 embedded entirely in one branch of T and T_2 embedded entirely in the other branch, giving $l(u_1, u_2) = l(u_1, \emptyset) + l(u_2, \emptyset)$, (2) with T_1 and the "half" of T_2 rooted at u_2^L embedded in one branch of T and the other half of T_2 embedded in the other branch of T, giving $l(u_1, u_2) = l(u_1, u_2^L) + l(\emptyset, u_2^R)$, (3-5) similarly to (2), (6,7) with half of each of the T_i embedded in each subtree of T, giving $l(u_1, u_2) = l(u_1^L, u_2^L) + l(u_1^R, u_2^R)$ or $l(u_1, u_2) = l(u_1^L, u_2^R) + l(u_1^R, u_2^L)$. The value of $l(u_1, u_2)$ is obtained as the minimum over these 7 possibilities. ∎

4 The Shortest Common Supersequence Problem for p-Sequences

We show $W[1]$-hardness for pSCS parameterized by the number of input sequences. Furthermore we show that the problem is fixed parameter tractable if we allow the supersequence to be a bit longer (by parameter r) than the size of the alphabet. We also give an FPT algorithm for the version of the problem where we only parameterize by r and have complete p-sequences as input.

4.1 Hardness Results

We define the following problem

p-Sequence Shortest Common Supersequence (pSCS I)

Input: p-sequences $x_1, ..., x_k$ and a positive integer M

Parameter: k. *Question:* Is there an rl-sequence x, with $|x| \leq M$ and $x_i \subseteq x$ for $i = 1, ..., k$?

Theorem 4. pSCS I *is (1) hard for $W[1]$ and (2) NP-complete.*

Proof (sketch): We reduce from the Clique problem which is proven $W[1]$-complete in [DF95b,DF98] when parameterized by the size of the clique set. The proof is rather lengthy and technical so we have choosen not to include it here. Interested readers are referred to [FHKS98] for the complete details. ∎

4.2 Tractable Parameterizations

We say that a sequence S contains r *duplication events* if S is not a p-sequence but the exactly r symbols need be removed from S to result in a p-sequence. We define a *duplication event* for trees in a similar manner except we must remove r leaves which result in a tree homeomorphic to a p-tree. We define the following problem

BOUNDED DUPLICATION SCS I FOR P-SEQUENCES (BDSCS I)
Input: A family of k p-sequences $x_i \in \Sigma^*$ for $i = 1, ..., k$ and a positive integer r representing the number of duplication events. We assume that $|\Sigma| = n$ and that each symbol of Σ occurs in at least one of the input sequences.
Parameter: (k, r)
Question: Is there a common rl-supersequence x of length at most $n + r$?

Theorem 5. BOUNDED DUPLICATION SCS I FOR P-SEQUENCES *is fixed-parameter tractable.*

Proof (sketch): We describe the argument in terms of a one-person game that provides a convenient representation of a supersequence of a set of p-sequences. At each move of the game, a symbol $a \in \Sigma$ is chosen, and those sequences in the collection that have a as their first symbol are modified by deleting the first symbol. We will call this an a-move. The game is completed when all of the symbols have been deleted from all of the sequences. It is straightforward to verify that a set of p-sequences has a common supersequence x of length m if and only if the game can be completed in m moves, where the sequence of symbols chosen for the moves of the game is x. When the game is played for the set of sequences x_i we may refer to the *current* x_i, meaning the suffix of x_i that remains at an understood (current) point in the game.
Suppose the symbol a is chosen for a move of the game. For each of the sequences x_i, one of the following statements must be true: (1) The symbol a is the first symbol of the current x_i and does not otherwise occur in the current x_i. (2) The symbol a does not occur in the current x_i. (3) The symbol a occurs in the current x_i, but is not the first symbol. If for an a-move of the game only (1) and (2) occur, we call this a *good* a-move. A move that is not good is *bad*. Our algorithm is based on the following claims (proofs omitted).
Claim 1. If for the input to the problem, a game is played that involves at least r bad moves, then the corresponding common supersequence x has length at least $n + r$. *Claim 2.* For any yes-instance of the problem, there is a game that can be completed with at most r bad moves.
We can now describe an FPT algorithm based on the method of search trees [DF95c]. By Claim 2, if the answer is "yes" then there is a game that completes with no more than r bad moves.
Algorithm. (0) The root node of the search tree is labeled with the given input. (1) A node of the search tree is expanded by making a sequence of good moves (arbitrarily) until no good move is possible. For each possible nontrivial bad move (i.e., one that results in at least one deletion), create a child node labeled with the set of sequences that result after this bad move. (2) If a node is labeled by the set of empty sequences, then answer "yes". (3) If a node has depth r in the search tree, then do not expand any further.
The correctness of the algorithm follows from Claim 2 and the following, which justifies that the sequence of good moves in (1) can be made in any order without increasing the length of the game.
Claim 3. If two different good moves, an a-move and a b-move, for $a, b \in \Sigma$, are

possible at a given point in a game, then there is a completion of the game in a total of l moves including m bad moves and starting with the a-move if and only if there is a completion of the game in a total of l moves including m bad moves and starting with the b-move. The proof is omitted due to space limitations. The running time of the algorithm is bounded by $O(k^r \cdot n)$. ∎

It is easy to see that we can extend this FPT-algorithm to the case that the input consists of rl-sequences instead of p-sequences.

Corollary 1. BOUNDED DUPLICATION SCS I *is fixed-parameter tractable.*

We next consider the question of what happens to the complexity of the above problem if the parameter is considered to be just r, rather than the pair (k, r). If we restrict the input sequences to be complete, then we can show fixed parameter tractability for a more general weighted problem. Indeed, as is frequently the case for FPT algorithms, solving this more general problem seems to be the key to proving tractability for the unweighted form of the problem.

BOUNDED DUPLICATION SCS II FOR COMPLETE P-SEQUENCES (BDCSC II)
Input: Complete p-sequences x_i over an alphabet Σ of size n, a positive integer r, and a cost function $c : \Sigma \to Z^+$.
Parameter: r
Question: Is there a common supersequence x of duplication cost $||x||_c \leq r$ where the duplication cost is defined as $||x||_c = \sum_{a\in\Sigma}(n_a(x) - 1)c(a)$ $n_a(x)$, $a \in \Sigma$, denotes the number of ocurrences of symbol a in x.

Theorem 6. BOUNDED DUPLICATION SCS II FOR COMPLETE P-SEQUENCES *is fixed-parameter tractable.*

Proof (sketch): Say that two symbols $a, b \in \Sigma$ are *combinable* if ab is a substring of each x_i. We argue that if $|\Sigma| > 3r+1$ then either there is a combinable pair of symbols or the answer is "no". Suppose the answer is "yes" and let x be any solution. Let x' be the subsequence of x consisting of those symbols of Σ that are not repeated in x. There is a natural factorization of x' into at most $2r+1$ substrings of x. Let Σ' be the set of symbols occuring in x'. Then $|\Sigma'| > 2r+1$ and at least one of the substrings of the factorization of x' must have length at least two. Let ab denote a substring of x of two symbols in Σ'. We argue that a and b are combinable. If not, then in at least one of the x_i we do not have the substring ab, and since the x_i are complete, one of the following must hold: (1) x_i contains the substring ba, (2) x_i contains a subsequence acb, or (3) x_i contains a subsequence bca. It is easy to check that in each of these cases it is impossible for x to be a supersequence of x_i.
The algorithm is described as follows, based on the method of problem kernels. The argument above shows that we can efficiently determine a reason to answer "no" (if $|\Sigma| > 3r+1$), or we can find a combinable pair ab of symbols. If we find a combinable pair, then we replace the substring ab in each x_i with a new symbol s_{ab} and give this symbol the duplication cost $c(s_{ab}) = c(a) + c(b)$. Given that the answer to this modified input is "yes" if and only if the answer for the

original input is "yes", this gives us a way of reducing to a problem kernel in time $O(N^2)$ where N is the total size of the input to the problem. The proof of correctness is omitted due to space limitations. ∎

It would be very nice to settle the parameterized complexity of BOUNDED DUPLICATION SCS III FOR P-SEQUENCES (BDSCS III), defined in exactly the same way as BDSDS I exept that the parameter is simply r rather than the pair (k, r). The proof of Theorem 2 shows the following

Theorem 7. *If BDSCS III is fixed parameter tractable, then so is DFVS.*

The interest of this theorem is that DFVS has resisted a number of attempts to settle its parametric complexity and is considered a major open problem [DF98], with most expert opinion favoring the conjecture that it is in *FPT*.

5 The Smallest Common Supertree Problem for Phylogenetic Trees

In computational biology the question arises how to resolve the species tree for a given set of gene trees such that the number of paralogous duplications is minimized. A *species tree* (or phylogenetic tree) is a rooted binary leaf labeled tree where the labels correspond to a set of taxa. Ma et al. [MLZ98] consider this question under the cost model introduced by Goodman et. al. [GCMRM79] and prove that the problem is *NP*-complete if the gene trees are rl-trees. An easy reduction leads to the *NP*-completeness of the same problem when the gene trees in the input are p-trees (proof omitted).

OPTIMAL SPECIES TREE I FOR P-TREES (pOST I)
Input: p-trees $G_1, \ldots, G_k$
Question: Does there exist a species tree S with minimum duplication cost $\sum_{i=1}^{k} m(G_i, S)$? (Here $m(G_i, S)$ is the number of duplications under the least common ancestor mapping of the gene tree G in the species tree S [MLZ98].)

Theorem 8. *pOST I is NP-complete.*

We consider in this section the problem of finding the smallest common supertree for a given set of p-trees (pSCT). We could interpret the resulting rl-tree in the following sense. A duplication node d in an rl-tree R is an internal node of R where the subtrees R_1, R_2 of R induced by the children d_1, d_2 of d have leaf labels in common. For given gene trees $G_1, \ldots, G_k$ we ask for the smallest rl-tree containing $G_1, \ldots, G_k$ as subtrees. Via the duplication nodes we can extract a p-tree (species tree) S which "explains" $G_1, \ldots, G_k$ in terms of the duplication nodes. That S is an approximation for pOST I is a consequence of the following consideration. The pSCT-problem asks for an rl-tree R containing $G_1, \ldots, G_k$. The minimum size of R gives a lower bound on the number of gene duplications. To see this, note that there can exist gene trees $G_i, G_j, i \neq j$, using the same duplication node, although two different gene duplications are represented. (We say a duplication node is *used* if the embedding of the gene tree in the rl-tree contains subtrees of the gene trees in both subtrees of the duplication node.) The exact number of gene duplications for $G_1, \ldots, G_k$ and R is $\sum_{i=1}^{k}$(minimum number of used nodes for an embedding G_i is embedded in R). Although pSCT is $W[1]$-hard (Theorem 9) the problem is fixed parameter tractable for complete p-trees if we ask for a common supertree of size at most $n+r$ for parameter pair (k, r).

5.1 Hardness Results

We define the following problem

Shortest Common Supertree for p-Sequences (pSCT I)
Input: binary p-trees $T_1, \ldots, T_k$ and a positive integer m
Parameter: k.
Question: Is there an rl-tree T, with $|T| \leq m$ and $T_i \subseteq T$ for $i = 1, \ldots, k$?

Noting that the construction in the second part of Theorem 2 also applies to pSCT I, we receive the following theorem.

Theorem 9. pSCT I *(1) hard for* $W[1]$ *and (2)* NP*-complete.*

5.2 Tractable Parameterizations

The parameterization of SCT that we next consider is defined as follows.

Bounded Duplication SCT for binary p-Trees (BDSCT)
Input: A family of k complete binary p-trees T_i with leaf label set Σ, $|\Sigma| = n$, and a positive integer r representing the number of duplication events.
Parameter: (k, r)
Question: Is there a common binary supertree T of the T_i of size at most $n + r$?

As discussed above the definition is reasonable for applications in the study of gene duplication events in the sense that both k and r may be small and the input trees complete when complete sequence data is available for all of the species under consideration.

Theorem 10. Bounded Duplication SCT for binary p-Trees *is fixed parameter tractable.*

Proof (sketch): The algorithm is based on a combination of the methods of search trees and reduction to a problem kernel described in [DF95c,DF98]. To a given tree T_i and $a \in \Sigma$, we associate an ordered partition $\pi_i(a)$ of $\Sigma - \{a\}$ in a natural way by considering the path $\rho_i(a) = (r_i = v_0, v_1, \ldots, v_s = l)$ in T_i from the root r_i or T_i to the leaf l labeled by a. Since the tree is binary, each vertex of this path other than l has another child v'_j for $j = 0, \ldots, s-1$. Let $T_i(j)$ denote the subtree rooted at v'_j and let $L_i(j)$ denote the set of leaf labels of the subtree $T_i(j)$. Then $\pi_i(a) = (L_i(0), \ldots, L_i(s-1))$.
We say that $a \in \Sigma$ is *good* if for all i, i', $1 \leq i, i' \leq k$, $\pi_i(a) = \pi_{i'}(a)$, and otherwise we say that a is *bad*. Note that if a is good then there is an obvious possibility of breaking the problem down into subproblems, since for each class

of the ordered partition $\pi(a)$ each T_i yields a subtree having precisely this set of leaf labels. We will refer to this as the subproblem *induced* by the class of the ordered partition.

The algorithm proceeds by attempting to find a good $a \in \Sigma$ (this is easy by just computing and comparing the partitions), and if a good a is found, then branching in a search tree by trying all possibilities for allocating the "budget" of k duplications among the nontrivial subproblems induced by classes of $\pi(a)$. Here by *nontrivial* we mean the situation where the induced subtrees of the subproblem are not all isomorphic, so that at least one duplication event is required for a common supertree. Note that determining whether a subproblem is trivial is easy. If there are more than k nontrivial subproblems then we answer "no".

We continue in this way to build a search tree until there are no more good symbols that allow further breakdowns of the problem. When the relevant subproblems are *small* (bounded in size by an appropriate function of k and r) then we answer by exhaustive search. The correctness and fixed parameter tractability of the procedure follows from the following two claims.

Claim 1. If there is a good $a \in \Sigma$ then an optimal solution can be computed by solving the subproblems and gluing these together along the path $\rho(a)$. The proof of this claim is nontrivial and omitted for reasons of length.

Claim 2. There is a function $f(r)$ such that if $|\Sigma| > f(r)$ and $k \geq 2$ and all $a \in \Sigma$ are bad, then the answer is "no".

The proof of this claim makes use of Bunemann's Theorem [B71] on the reconstructibility of p-trees from triples, and on being able to find more than k disjoint sets of conflicting triples when $|\Sigma|$ is large and all $a \in \Sigma$ are bad. ∎

6 Open Problems

Our main contribution has been to explore the complexity of basic problems about sets of symbols structured by binary trees and sequences. These are related to the MAST problem, which has an important, nontrivial polynomial-time algorithm, a situation that suggests that the other three problems that are naturally similar to MAST might also be solvable in polynomial-time. One of these, the sequence analog of MAST, we have shown to be polynomial, while the problems dual to MAST concerning common supersequences and supertrees remain NP-complete. One interesting aspect of our results are the strong parallels exhibited between the complexity of tree and sequence problems.

A number of open problems remain. In particular: (1) The fixed parameter tractability results are presently slightly stronger for sequences. We conjecture that tree analogs of Theorems 5 and 6 should hold. (2) Does Theorem 6 still hold if the input sequences are not required to be complete? This seems to be a reasonable conjecture, but we have shown that this would imply that Directed Feedback Vertex Set is in *FPT*, also a reasonable conjecture, but apparently a difficult one [DF98]. (3) Because in biological applications the p-sequences and p-trees are often complete, the complexity of the four basic problems LCS, SCS, LCT, and SCT for complete p-objects are interesting.

(4) Can a guaranteed performance bound be establish for the application of Theorem 10 to problems of paralogous gene duplications discussed in Section 5? A possible heuristic for computing parsimonious annotations is to: (i) Compute an rl-supertree T of minimum size using the algorithm of Theorem 10, and then (ii) Extract an annotated p-tree from T.

References

[B97] D. Bryant. "Building Trees, Hunting for Trees, and Comparing Trees – Theory and Methods in Phylogenetic Analysis," Ph.D. Thesis. Department of Mathematics. University of Canterbury, 1997.

[B71] P. Buneman. "The Recovery of Trees from Measures of Dissimilarity," In F. R. Hodson, D. G. Kendall, P. Tautu, editors, *Mathematics in the Archaeological and Historical Sciences*, pp. 387–395. Edinburg University Press, Edinburgh, 1971.

[DF93] R. Downey and M. Fellows. "Fixed Parameter Tractability and Completeness III: Some Structural Aspects of the W-Hierarchy," in: K. Ambos-Spies, S. Homer and U. Schöning, editors, *Complexity Theory: Current Research*, Cambridge Univ. Press (1993), 166–191.

[DF95a] R. G. Downey and M. R. Fellows. "Fixed Parameter Tractability and Completeness I: Basic Theory," *SIAM Journal of Computing* 24 (1995), 873-921.

[DF95b] R. G. Downey and M. R. Fellows. "Fixed Parameter Tractability and Completeness II: Completeness for $W[1]$," *Theoretical Computer Science A* 141 (1995), 109-131.

[DF95c] R. G. Downey and M. R. Fellows. "Parametrized Computational Feasibility," in: *Feasible Mathematics II, P. Clote and J. Remmel (eds.)* Birkhauser, Boston (1995) 219-244.

[DF98] R. G. Downey and M. R. Fellows. *Parameterized Complexity*, Springer-Verlag, 1998.

[FPT95] M. Farach, T. Przytycka, and M. Thorup. "On the agreement of many trees" *Information Processing Letters 55* (1995), 297–301.

[FHKS98] M. R. Fellows, M. T. Hallett, C. Korostensky, U. Stege. "The complexity of problems on sequences and trees." Technical Report, ETH-Zurich, 1998.

[GJ79] M. R. Garey and D. S. Johnson. " Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman, San Francisco, 1979.

[GCMRM79] 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," *Syst. Zool.*(1979), 28, 132–163.

[GMS96] R. Guigó, I. Muchnik, and T. F. Smith. "Reconstruction of Ancient Molecular Phylogeny," *Molecular Phylogenetics and Evolution* (1996),6:2, 189–213.

[MLZ98] B. Ma, M. Li, and L. Zhang. "On Reconstructing Species Trees from Gene Trees in Term of Duplications and Losses," *Recomb 98*, to appear.

[M78] D. Maier. "The Complexity of Some Problems on Subsequences and Supersequences," *J. ACM*, 25,2(1978), 322–336.

[P94] R. D. M. Page. "Maps between trees and cladistic analysis of historical associations among genes, organisms, and areas," *Syst. Biol. 43* (1994), 58–77.

[P97] T. Przytycka. private communication, 1997.

[Z97] L. Zhang. "On a Mirkin-Muchnik-Smith Conjecture for Comparing Molecular Phylogenies," *Journal of Computational Biology* (1997) 4:2, 177–187.