Phylogenetic Tree Reconstruction with Protein Linkage

Junjie Yu[1], Henry Chi Ming Leung[1], Siu Ming Yiu[1], Yong Zhang[1], Francis Y.L. Chin[1], Nathan Hobbs[2], and Amy Y.X. Wang[2]

[1] Department of Computer Science, The University of Hong Kong
{jjyu,cmleung2,smyiu,yzhang,chin}@cs.hku.hk
[2] Institute for Interdisciplinary Information Sciences, Tsinghua University
marscheese@gmail.com, amywang@mail.tsinghua.edu.cn

Abstract. When reconstructing a phylogenetic tree, one common representation for a species is a binary string indicating the existence of some selected genes/proteins. Up until now, all existing methods have assumed the existence of these genes/proteins to be independent. However, in most cases, this assumption is not valid. In this paper, we consider the reconstruction problem by taking into account the dependency of proteins, i.e. protein linkage. We assume that the tree structure and leaf sequences are given, so we need only to find an optimal assignment to the ancestral nodes. We prove that the Phylogenetic Tree Reconstruction with Protein Linkage (*PTRPL*) problem for three different versions of linkage distance is NP-complete. We provide an efficient dynamic programming algorithm to solve the general problem in $O(4^m \cdot n)$ and $O(4^m \cdot (m+n))$ time (compared to the straight-forward $O(4^m \cdot m \cdot n)$ and $O(4^m \cdot m^2 \cdot n)$ time algorithm), depending on the versions of linkage distance used, where n stands for the number of species and m for the number of proteins, i.e. length of binary string. We also argue, by experiments, that trees with higher accuracy can be constructed by using linkage information than by using only hamming distance to measure the differences between the binary strings, thus validating the significance of linkage information.

Keywords: Phylogenetic tree reconstruction, Protein linkage, NP-complete.

1 Introduction

Discovering evolutionary relationships among species is an important problem in bioinformatics. Given a set of species, the phylogenetic tree reconstruction problem is to reconstruct an evolutionary tree where each leaf of the tree represents an input species and each internal node u represents a hypothetical ancestor of the species represented by all leaf nodes in the subtree rooted at u. There are two common representations of the species in this problem: a DNA (or protein) sequence [1] of some selected genes (or proteins) which are believed to be relevant to the evolution process, or a binary string [2] that indicates the existence/absence of those selected genes (or proteins) in the species. Based on these representations, there are three general approaches for the reconstruction of a phylogenetic tree: (1) distance methods, (2) parsimony methods, and (3) likelihood methods.

L. Bleris et al. (Eds.): ISBRA 2012, LNBI 7292, pp. 315–327, 2012.

Distance methods first define an evolutionary distance measure between two DNA sequences (or two binary strings) that represent the two species being compared. A distance matrix that captures the evolutionary distance between each pair of species is used as an input to the phylogentic tree construction problem. The distance method then infers a phylogenetic tree from the distance matrix by grouping closer species together in a subtree. The most well-known distance method is the Neighbor-joining (NJ) algorithm [3], which was later improved in terms of time complexity by the Fast Neighbor Joining (FNJ) algorithm [4]. Other distance methods such as ProfDists [5], Profile Neighbor Joining [6], and Weighbor [7], are efficient algorithms with polynomial time complexities, but they lack robustness. Moreover, the effectiveness of these methods relies very much on the evolutionary distance measure being used.

Maximum parsimony methods aim to find a phylogenetic tree which can explain the given sequences with a minimum number of substitutions. This problem is known to be NP-hard [8]. Heuristic methods, such as the genetic algorithm [9], tabu search [10], simulated annealing [11], and hill climbing [12] were introduced to find near optimal results. If the tree structure is known (usually referred to as the "small parsimony problem"), the problem can be solved in polynomial time using the Fitch-Hartigan algorithm [13] or Sankoff algorithm [14].

Likelihood methods (sometimes called probabilistic methods) are based on evolutionary models, such as the gene evolution model [15], Jukes-Cantor's one-parameter model, or Kimura's two-parameters model [16]. Maximum likelihood methods [17-18] and Bayesian methods [19] are examples of probabilistic methods that find phylogenetic trees by maximizing the likelihood (or posterior probability) of generating the observed sequences. Probabilistic methods and parsimony methods are usually more robust but are more expensive computationally.

There are other methods for which the input is a set of small subtrees, each of which captures the evolutionary relationship of three (called triplets) or four (called quartets) species among the given set of species. The problem is to combine these subtrees into a single phylogenetic tree such that the evolutionary relationship of the species is consistent with the small subtrees. However, this problem has been shown to be difficult. Even when all given quartets are consistent and can be combined to form a phylogenetic tree, finding such a tree is still NP-complete [20]. There are heuristic methods [21-22] and PTAS [23-24] algorithms for the case when all $\binom{n}{4}$ quartets are given. Another issue for this approach is that the availability of the triplets and quartets is usually limited. In this paper, we do not consider this type of input but focus on the representation of using a binary string to indicate the existence of selected genes/proteins.

In distance methods and maximum parsimony methods where binary strings are input, a common assumption is that each bit of the binary string is independent, i.e. the existence of a particular gene/protein in a species is independent of the existence of another gene/protein in the same species. This assumption is obviously an oversimplification because genes/proteins in real life usually work together with other genes/proteins in various ways, e.g. in metabolic pathways [25], protein complexes [26], and signal transduction pathways [27]. Thus, the existence of different genes in a species may be dependent [28], and when reconstructing a phylogenetic tree, this kind of dependence should be considered.

There is strong evidence [28] that some proteins/genes should co-exist, which is referred to as protein linkage. From an evolutionary point of view, these functionally dependent proteins should usually be present (or absent if the function is no longer needed) in the same generation. Based on the principle of parsimony, in addition to finding a phylogenetic tree with the minimum number of insertions/deletions (hamming distance), it is not desirable to have some proteins present and others absent if they belong to the same functional group. To capture this biological property, we introduce a new model for calculating the cost of a phylogenetic tree by incorporating protein linkage information using the parsimony approach. Given a tree topology, we define the *phylogenetic tree reconstruction with protein linkage problem* (the *PTRPL* problem) as finding an assignment of internal nodes which minimize the total number of protein insertions/deletions (hamming distance) and different versions of linkage cost. We show that the *PTRPL* problem under three different versions of linkage costs resolve to be NP-complete.

Although the versions of the problem we consider are NP-complete, we believe that linkage information is important for evolutionary studies. To further this area of research, we provide an efficient dynamic programming algorithm to find an optimal assignment in $O(4^m \cdot n)$ or $O(4^m \cdot (m+n))$ time when the tree topology is given (compared to the straight-forward brute-force $O(4^m \cdot m \cdot n)$ or $O(4^m \cdot m^2 \cdot n)$ time algorithms), where n is the number of species and m is the length of the binary string for each species. We demonstrate the effectiveness of our algorithm on real biological data that contain protein and protein linkage information. We incorporate our algorithm into the Nearest Neighbor Interchange algorithm [33]. For every step of the Nearest Neighbor Interchange, based on the current topology, we compute the minimum cost using our approach. The results show that we are able to reconstruct more accurate phylogenetic trees (with 11% higher in accuracy) than other existing approaches that ignore linkage information. The running time of our algorithm is acceptable for some practical sizes of m and is about 5 – 35 times faster than the brute-force algorithm according to our experimental results.

2 Different Versions of Linkage Distance

The cost of each evolution step depends on the number of protein insertions/deletions (hamming distance) and the degree of protein association (protein linkage). Given two length m binary strings, u and v, that represent the existence of m proteins in two species, the cost between u and v is the sum of the hamming distance and the protein linkage distance (described in the following subsections) between u and v. Given a tree T with n leaf nodes, each labeled with a binary string of length m, and the protein linkage information of these m proteins, the *PTRPL* problem is to assign length-m binary strings to the internal nodes of T that minimize the total cost of the tree T, i.e. the sum of the cost of all edges in T. We define different versions of linkage distance according to various assumptions about evolution and show that the *PTRPL* problem is NP-complete under each of the versions of linkage distance we define.

2.1 Whole Block Linkage Distance

Whole block linkage distance assumes that all proteins should be present or absent in the species as a block (consistent state) during evolution. From this presence or absence assumption, blocks are usually disjoint, i.e. each protein normally belongs to at most one block. The linkage cost will be w if the species evolves from a consistent state to an inconsistent one and zero otherwise. For all $\sigma, \sigma' \in \{0,1\}^m$, and $w > 0$,

$$l_{wb}(\sigma, \sigma') = \begin{cases} w & \sigma \in \{0^m, 1^m\} \text{ and } \sigma' \notin \{0^m, 1^m\} \\ 0 & else \end{cases}$$

The NP-complete proof is by reduction from the NP-complete NOT-ALL-EQUAL 3SAT problem [32], defined as: given a Boolean formula $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_t = (x_{11} \vee x_{12} \vee x_{13}) \wedge (x_{21} \vee x_{22} \vee x_{23}) \wedge \cdots \wedge (x_{t1} \vee x_{t2} \vee x_{t3})$ over t clauses and k variables, the problem is to find a truth assignment for the variables such that each clause C_i of ϕ has at least one true literal and one false literal, $1 \leq i \leq t$.

The NP-complete proof is to transform ϕ into an instance of the *PTRPL* problem such that ϕ is satisfiable if and only if the *PTRPL* instance has an internal node assignment that makes total distance of the tree at most $(2k + 15)t + (2k + 2)k$.

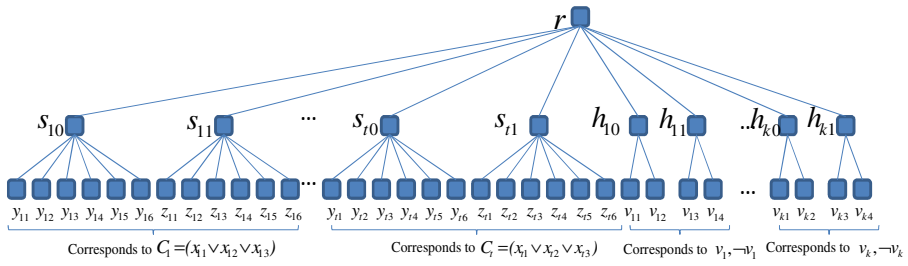

Fig. 1. PTRPL instance from the reduction of NOT-ALL-EQUAL 3SAT problem

The *PTRPL* instance consists of a 3-level tree T (Figure 1) with each node represented by a length-$2k$ binary string, one bit for each variable or its negation, where all $2k$ bits are considered to be one block. Root r has $(2t + 2k)$ children with s_{i0} and s_{i1} corresponding to clause C_i in ϕ, $1 \leq i \leq t$ (satisfaction testing), and h_{j0} and h_{j1} corresponding to variable v_j, $1 \leq j \leq k$ (truth setting). Node s_{i0} (together with s_{i1} representing C_i) has 6 children (leaf nodes) whose binary strings represent 6 different satisfiable assignments (100, 010, 001, 110, 101, 011) on its 3 literals x_{i1}, x_{i2}, x_{i3} at the corresponding bit positions of the $2k$-binary string, and 0 at all the other $(2k-3)$ bits. The binary strings for the children of s_{i1} are similar to that of s_{i0}, except that all the other $(2k - 3)$ bits are 1s (Figure 2). Thus, in order to have the total minimum hamming distance $(2k-3)$ and 18 from s_{i0} and s_{i1} to r and their leaf nodes respectively, $s_{i0}(s_{i1})$ would have 0's (1's) at these $(2k-3)$ bit positions and the same bit assignments as root r at the remaining three bit positions (for clause satisfaction). Similarly, both h_{j0} and h_{j1} each have 2 children that encode the truth assignment for v_j and $\neg v_j$ by having (10, 01) at the

corresponding two bit positions of the $2k$-binary string, with all other $(2k–2)$ bits assigned 0 or 1 for the children of h_{j0} or h_{j1} respectively. Again, h_{j0} or h_{j1} would have 0 or 1 at the $(2k–2)$ bit positions and the same bit assignments as root r at the two bit positions for the truth assignments of v_j and $\neg v_j$. When the internal nodes are assigned in this way, the total hamming distance is minimum, i.e. $(2k + 15)t + (2k + 2)k$. In order to have zero linkage distance, none of the internal nodes can be 0^{2k} or 1^{2k}, i.e. both h_{j0} and h_{j1} ensure that either v_j or $\neg v_j$ (but not both) is assigned 1, and s_{i0} (resp. s_{i1}) ensure that not all the literals in clause C_i are assigned 0's (resp. 1's) (the NOT-ALL-EQUAL 3SAT problem). The binary string of root r would contain an encoding of the truth assignment of each variable, and each internal node (with the exception of r) would have the same binary string as one of its children, i.e. the clause C_i is satisfiable iff ϕ is satisfiable.

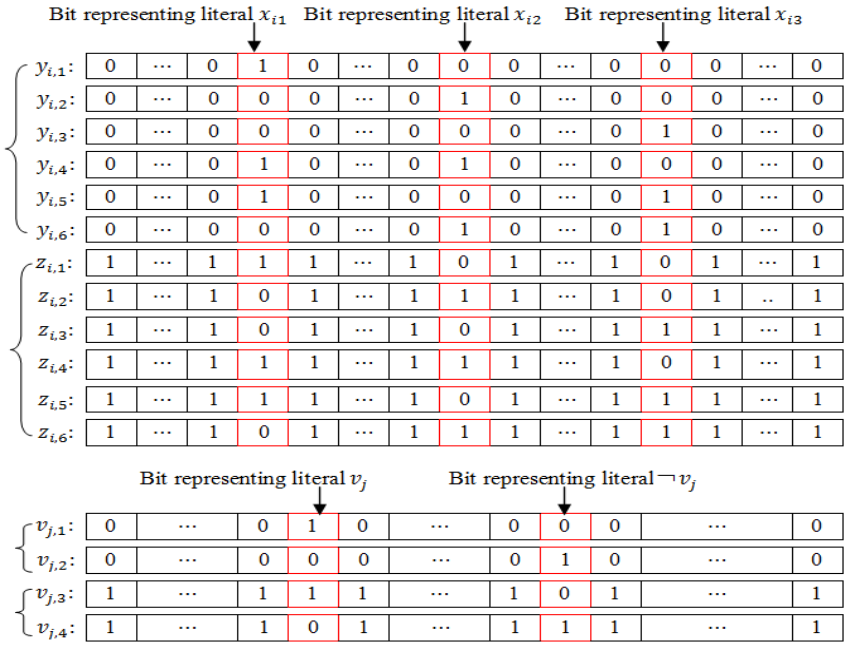

Fig. 2. The binary string for leaf nodes

2.2 Partial Block Linkage Distance

The only difference between the whole block and partial block linkage distance is the addition of a cost when a binary string evolves from an inconsistent state to another inconsistent state (as a state transition of this kind does not make any progress in evolution). For all $\sigma, \sigma' \in \{0,1\}^m$, $w > 0$ and $0 < \delta \leq w$:

$$l_{pb}(\sigma,\sigma') = \begin{cases} 0 & \sigma' \in \{\sigma, 0^m, 1^m\} \\ \delta & \sigma' \notin \{\sigma, 0^m, 1^m\} \text{ and } \sigma \notin \{0^m, 1^m\} \\ w & \sigma' \notin \{0^m, 1^m\} \text{ and } \sigma \in \{0^m, 1^m\} \end{cases}$$

The *PTRPL* problem is also NP-complete under partial block linkage distance. However, the construction of the problem instance from ϕ as shown in the NP-complete proof for the whole block linkage distance might not work for partial block linkage distance, because some decreases in linkage distance might compensate for increases in hamming distance and the resulting assignment might not have to achieve a minimum hamming distance. For example, it is possible that the binary string at r adopts one of its children's binary strings, so as to reduce one linkage distance at the expense of its hamming distances with other children. In order to prevent such scenario, one can extend each binary string by $2(k+t)A$ bits with a different section of A 1-bits, corresponding to each child of r: s_{i0}, s_{i1}, h_{j0} and h_{j1}, $1 \le i \le t$ and $1 \le j \le k$ (and all other bits 0's). We determine the value of A such that hamming distance will dominate the linkage distance by having $A > w(14t + 6k)$ where the total number of tree edges is$14t + 6k$. For example, the last $2(k+t)A$ bits of $s_{10}{}'$s children are $1^A 0^A \cdots 0^A$; the last $2(k+t)A$ bits of $s_{11}{}'$schildren are $0^A 1^A 0^A \cdots 0^A$; $0^A 0^A 1^A 0^A \cdots 0^A$ for $s_{20}{}'$s children, etc. In this case, the root would have an assignment of 0's as its last $2(k+t)A$ bits in order to minimize the total hamming distance to all its $2(k+t)$ children. With this assignment of the internal nodes, the total cost (hamming distance plus linkage distance) will be minimum iff ϕ is satisfiable. Formally, the hamming distance would be $(2k+15)t + (2k+2)k + (2t+2k)A$, and the linkage distance would be $[2(k+t) + 10k + 2t]\delta$.

Note that after the extension of the $2(k+t)A$ bits, the minimum hamming distance can still be achieved if the first $2k$-bits of some internal nodes are assigned 0's or 1's (i.e. some clauses are not satisfiable with the same true/false assignment for all 3 literals or some literals have the same truth assignment as their negations). However, assignments of this kind are avoided because the linkage distance between internal nodes to their children would be increased by at least δ.

2.3 Pairwise Linkage Distance

Pairwise linkage distance is defined through a set P of protein pairs, where two proteins in a protein pair are expected to be present/absent at the same time. In contrast to block linkage, we assume that each protein can belong to more than one pairwise linkage. Note that the whole block and partial block linkage distance can be partially modeled by pairwise linkage distance by having all proteins in a block paired to each other.

Let P be a set of protein pairs (p, q), where p and q represent two bit positions in the binary string, the linkage distance=$\sum_{(p,q)\in P} l_p(\sigma_{pq}, \sigma'_{pq})$ where σ_{pq} (σ'_{pq}) = the two bits at positions p and q of binary string $\sigma(\sigma')$, with $w > 0$, and $0 \le \delta \le w$,

$$l_p(\sigma_{pq}, \sigma'_{pq}) = \begin{cases} 0 & \sigma'_{pq} \in \{\sigma_{pq}, 00, 11\} \\ \delta & (\sigma_{pq}, \sigma'_{pq}) \in \{(01,10),(10,01)\} \\ w & \sigma_{pq} \in \{00,11\} \text{ and } \sigma'_{pq} \in \{01,10\} \end{cases}$$

The NP-complete proof is by reduction from the NP-complete (3, 4)-SAT problem [29-30], which is a special case of the 3SAT problem, in which each variable appears exactly four times in ϕ.

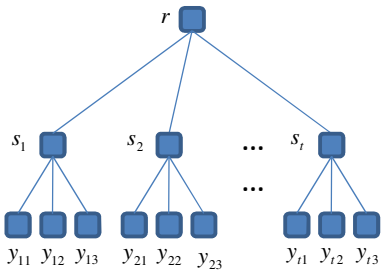

Fig. 3. 3-level tree for reduction

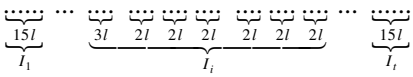

Fig. 4. Binary string of a leaf node

Given a formula ϕ with t clauses, we construct a 3-level tree T (Figure 3) for the *PTRPL* instance where root r has t children s_i, corresponding to clause C_i in ϕ, $1 \le i \le t$, and s_i has three children y_{i1}, y_{i2}, y_{i3}, corresponding to literals x_{i1}, x_{i2}, x_{i3} in the clause $C_i = (x_{i1} \vee x_{i2} \vee x_{i3})$.

Each leaf node is represented by a length-$15lt$ binary string that consists of t length-$15l$ sections with the i-th section denoted by I_i, corresponding to clause $C_i, 1 \le i \le t$ (Figure 4). Note that we determine the value of l such that the linkage distance dominates the hamming distance; formally, $l > [(5t+3) \cdot t/(4w)]$. The first $3l$ bits in I_i are a dummy section denoted by $I_i(dummy)$, used to balance linkage distance to guarantee strings assigned to s_i same as those of one of its three leaf children when ϕ is satisfiable. The other 6 length-$2l$ literal sections represent the literals or their negations in clause C_i; they are denoted by $I_i(x_{i1})$, $I_i(x_{i2})$, $I_i(x_{i3})$, $I_i(\neg x_{i1})$, $I_i(\neg x_{i2})$, $I_i(\neg x_{i3})$. For leaf node y_{i1} (resp. y_{i2} and y_{i3}), only the length-$3l$ $I_i(dummy)$ section and length-$2l$ $I_i(x_{i1})$ (resp. $I_i(x_{i2})$ and $I_i(x_{i3})$) are assigned 1's; the remaining $(15lt - (3l + 2l))$ bits are assigned 0's. P is defined in such a way so as to enforce satisfaction testing of each clause and consistent truth testing for variables. Set P contains three types of protein pairs: *T-1*, *T-2*, and *T-3*:

1) *T-1* protein linkage: between each bit of $I_i(dummy)$ to each bit of $I_i(x_{i1})$, $I_i(x_{i2})$, $I_i(x_{i3})$, $I_i(\neg x_{i1})$, $I_i(\neg x_{i2})$, $I_i(\neg x_{i3})$; $1 \le i \le t$.
2) *T-2* protein linkage: between each pair of bit for sections $I_i(x_{i1})$ and $I_i(x_{i2})$, $I_i(x_{i1})$ and $I_i(x_{i3})$, $I_i(x_{i2})$ and $I_i(x_{i3})$; $1 \le i \le t$.
3) *T-3* protein linkage: between each bit in section $I_i(v)$ and each bit in section $I_j(\neg v)$ where literals v or $\neg v$ occur in clause C_i, C_j; $1 \le i, j \le t, i \ne j$.

Since each variable appears exactly four times, there are an equal number of *T-1* and *T-2* linkages. When the strings for section I_i of r and s_i are the same and the remaining sections of s_i are all 0's, no *T-1* or *T-2* linkage costs occur between r and s_i. The subsections that represent variable v and its negation $\neg v$ in the two sections of r that correspond to the two different clauses cannot be assigned 1's at the same time; otherwise, the *T-3* linkage distance between r to its children would increase by at least $4wl^2$. Thus, when formula ϕ is satisfiable, s_i will have the same assignment as one of its leaf children and no *T-1*, *T-2* and *T-3* linkage costs will exist between r and s_i. When formula ϕ is not satisfiable, either some clause C_i cannot be satisfied, or some variable and its negation are both assigned 1s at the same time. In either case, the total linkage distance will increase by at least $4wl^2$. Therefore, we can prove that the total cost of the resulting tree of the *PTRPL* instance is at most $(5t+3)\cdot lt + 4(11w+2\delta)l^2t$ if and only if the formula in the *(3, 4)-SAT* problem is satisfiable, where t is the number of clauses in the *(3, 4)-SAT* problem, $l > [(5t+3)\cdot t/(4w)]$, $w > 0$ and $0 \le \delta \le w$.

3 General Algorithm for the *PTRPL* Problem

In this section, we introduce an algorithm for solving the *PTRPL* problem through the use of dynamic programming.

Similar to Sankoff algorithm [14] which considers each protein independently, we define $C(u,\sigma)$ to be the minimum total cost (including protein linkage distance) of the subtree rooted at node u when node u is assigned string σ, $\sigma \in \{0,1\}^m$, m is the maximum number of proteins being considered, i.e. the length of the binary strings. The value of $min_{\sigma\in\{0,1\}^m} C(r,\sigma)$, where r is the root of the known leaf-labeled phylogenetic tree *T*, gives the minimum cost of tree *T*. There are two basic cases for a leaf node u: $C(u,\sigma) = 0$ if u is represented by binary string σ, and $C(u,\sigma) = \infty$ if u is not represented by σ. The value of $C(u,\sigma)$ for all internal nodes can be calculated by the following recursion:

$$C(u,\sigma) = \sum_{v\in\, children\ of\ u} \min_{\sigma'\in\{0,1\}^m} \{d(\sigma,\sigma') + C(v,\sigma')\},$$

where $d(\sigma,\sigma')$ is sum of hamming distance and linkage distance between σ and σ'. Since there are at most $2^m \cdot 2^m = 4^m$ combinations of σ and σ' to consider, and each $d(\sigma,\sigma')$ takes $O(m)$ time to calculate the hamming distance plus whole/partial block linkage distance, and $O(m^2)$ time to calculate the hamming distance plus pairwise linkage distance, the total takes at most $O(4^m \cdot m \cdot n)$ or $O(4^m \cdot m^2 \cdot n)$ time for a tree with $O(n)$ edges. However, we can optimize the algorithm by precomputing $d(\sigma,\sigma')$ ahead of time for all combinations of σ and σ'. Each $d(\sigma,\sigma')$ can be calculated in $O(1)$ or $O(m)$ time depending on block and pairwise linkage, respectively (by considering each σ' with a particular σ one by one according to their gray code such that two successive binary strings differ by one bit, after exhausting all 2^m different values of σ', the next σ is considered according to its gray code), as each protein can relate to at most one block (resp. m

pairs) of linkage(s), the values at $d(\sigma,\sigma')$ for all pairs of σ and σ' will take no more than $O(4^m)$ (resp. $O(4^m \cdot m)$) time. As the value of $d(\sigma,\sigma') + C(v,\sigma')$ for each child *v* of *u* can be computed independently, we can perform the calculation edge by edge along with the values of $C(u,\sigma)$ for all internal nodes in $O(4^m \cdot n)$ time, where *n* is the number of leaf nodes. The time complexity for finding $min_{\sigma\in\{0,1\}^m} C(r,\sigma)$ is $O(4^m \cdot n)$ and $O\big(4^m \cdot (m+n)\big)$ for block and pairwise linkage respectively. Algorithm 1 gives the pseudocode of the algorithm.

Algorithm 1: *PTRPL* problem

Input: Phylogentic tree T with n leaf nodes, each leaf node u labeled with a binary sequence σ_u of length m.

Output: Minimum total cost (hamming distance and linkage distance) of *T*.

1: Pre-compute $d(\sigma,\sigma')$, for all pair $\sigma,\sigma' \in \{0,1\}^m$

2: Calculating the value of $min_{\sigma\in\{0,1\}^m} C(r,\sigma)$, r is the root of *T*, by the following recursion:

$$C(u,\sigma)=\begin{cases} 0 & u \text{ is a leaf and } \sigma = \sigma_u \\ \infty & u \text{ is a leaf and } \sigma \neq \sigma_u \\ \sum_{v\in u's\ children} min_{\sigma'\in\{0,1\}^m}\{d(\sigma,\sigma') + C(v,\sigma')\} & u \text{ is not a leaf} \end{cases}$$

4 Experiments and Results

In order to evaluate the significance of the use of linkage information in constructing a phylogenetic tree, we downloaded 10 phylogenetic trees from NCBI for experimentation. Each tree contains 10 species: 9 of them bacteria in 3 to 4 different genera and 1 Saccharomyces cerevisiae (yeast) working as an outlier. A total of 10 orthology groups were selected for each tree such that each species is represented by a length-10 binary string with a particular position in the binary string assigned "1" if and only if the species has a protein from the corresponding orthology group. Two orthology groups (positions) are considered as having a pairwise linkage if and only if the two orthology groups are either both present or both absent in at least 8 species. A set of orthology groups are considered to be in the same block if all pairs of orthology groups in the set have pairwise linkage with each other.

Nearest Neighbor Interchange algorithm [33] is applied to reconstruct the phylogenetic tree for the ten species based on different cost functions, including pairwise linkage distance, whole block linkage distance, partial block linkage distance and hamming distance. Given an initial tree topology, the Nearest Neighbor Interchange algorithm greedily updates the tree topology to minimize the total cost based on the corresponding cost function of the tree until no more improvement can occur. Since the initial tree topology has a dramatic effect on the final result, we randomly picked 100 initial trees for each experiment to minimize this effect. Among the 100 final results, the tree with the minimum total cost was considered as the

predicted tree. Different values of w and δ are considered, and the resulting assignemnts is not sensitive to the values of w and δ. We use $w = 1$ for the experiments of the whole block linkage distance problem and $w = 1$, $\delta = 0.5$ for the partial block linkage distance and pairwise linkage distance problem. These parameters gave slightly better results than other parameters.

We calculate the number of triplets in each predicted tree that match the NCBI tree. A triplet of three species A, B and C, where species A and B closer to each other than species C in the predicted tree, is considered as a match with the NCBI tree if a) species A and B are closer to each other than species C in the NCBI tree, or b) species A, B and C have the same closest ancestor in the NCBI tree.

Table 1. Percentage of correct triplets for different cost functions

	Pairwise linkage distance	Whole block linkage distance	Partial block linkage distance	Hamming distance
Average number of correct triplets	88.8	82	81.3	75.5
Percentage of correct triplets	74%	68.3%	67.8%	62.9%
Average running time	30 seconds	5 seconds	5 seconds	1 second

From the results shown in Table 1, the trees we predict using both hamming distance and linkage distance information are more accurate than the trees we predict using hamming distance only. Our conclustion is that protein linkage provides information for the reconstruction of a more accurate phylogentic tree. When we compare the trees we construct based on whole block linkage distance and partial block linkage distance, we find that the trees we construct using pairwise linkage distance are more accurate.

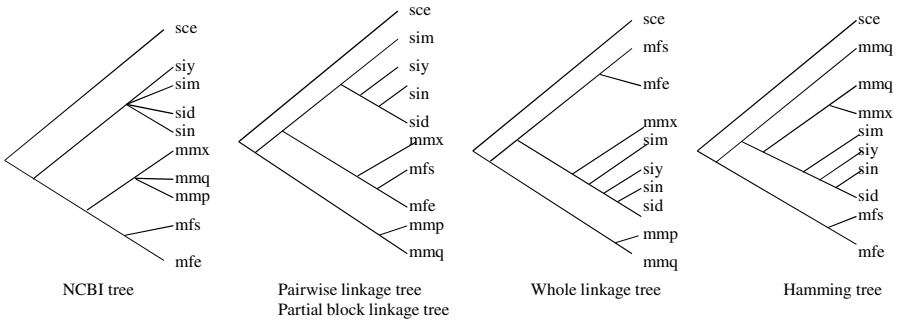

Fig. 5. NCBI tree topology and the predicted tree topologies based on pairwise linkage distance, whole block linkage distance, partial block linkage distance and hamming distance

Figure 5 shows an example of the NCBI tree topology and the predicted tree topologies based on pairwise linkage distance, whole block linkage distance, partial block linkage distance and hamming distance for a set of species (1 Saccharomyces, 2 Methanocaldococcus, 3 Methanococcus and 4 Sulfolobus). The NCBI tree and the predicted tree based on pairwise linkage and partial block linkage distance are similar except that species Methanococcus maripaludis C6 (mmx) in the Methanococcus is misclassified as Methanocaldococcus (70.8% of triplets are correct). The predicted tree based on whole block linkage distance also resembles the NCBI tree except that species mmx in the Methanococcus is misclassfied as Sulfolobus (65.8% of triplets are correct). However, the predicted tree based on hamming distance merges the Methanococcus and Sulfolobus incorrectly and is quite different from the NCBI tree (only 63.3% of triplets are correct).

All experiments were performed on a server machine with eight 2.4 GHZ CPUs and 140G memory. However, only one CPU is used in our experiments and the memory consumption for the program is less than 80G. The running times of the Neighbor Interchange algorithm based solely on hamming distance, hamming distance plus pairwise linkage distance and hamming distance plus block linkage distance on one experiment takes 1 second, 30 seconds and 5 seconds for a single experiment, respectively. Although the algorithm using protein linkage information requires a longer running time, it is acceptable for phylogenetic trees with a small set of species.

5 Conclusion

We have proved the NP-completeness for Phylogenetic Tree Reconstruction with Protein Linkages (*PTRPL*) problem under three definitions of linkage distance. A general algorithm for the *PTRPL* problem is presented for different linkages with time complexity $O(4^m \cdot n)$ and $O(4^m \cdot (m+n))$ for two variations of linkage distance (block linkage and pairwise linkage).

Lastly, we conducted experiments to confirm our hypothesis that the use of linkage information could lead to more accurate phylogenetic trees. For future work, we will try to further evaluate the efficiency and effectiveness of our dynamic programming algorithm. In particular, we shall study the *PTRPL* problem when the phylogenetic tree structure is not given, aim at developing effective heuristics, explore the linkage structures of real world data and develop algorithms for different versions of linkage structures.

Acknowledgements. This work was supported by grant HKU 7116/08E, HKU 719709E, NSFC 11171086.

References

1. Wang, L.-S., Leebens-Mack, J., Wall, P.K., Beckmann, K., Pamphilis, C.W., Warnow, T.: The Impact of Multiple Protein Sequence Alignment on Phylogenetic Estimation. Computational Biology and Bioinformatics 8, 1108–1119 (2011)

2. Zhou, Y., Wang, R., Li, L., Xia, X., Sun, Z.: Inferring Functional Linkages between Proteins from Evolutionary Scenarios. Journal of Molecular Biology 359 (2006)
3. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular Biology and Evolution 4, 406–425 (1987)
4. Elias, I., Lagergren, J.: Fast neighbor joining. Theoretical Computer Science (2008)
5. Wolf, M., Ruderisch1, B., Dandekar1, T., Schultz1, J., Müller, T.: ProfDistS (profile-) distance based phylogeny on sequence-structure alignments. Bioinformatics 24 (2008)
6. Muller, T., Rahmann, S., Dandekar, T., Wolf, M.: Accurate and robust phylogeny estimation based on profile distances: a study of the Chlorophyceae (Chlorophyta). BMC (2004)
7. Bruno, W.J., et al.: Weighted Neighbor Joining: A Likelihood-Based Approach to Distance-Based Phylogeny Reconstruction. Molecular Biology and Evolution (2000)
8. Foulds, L.R., Graham, R.L.: The Steiner Problem in Phylogeny is NP-Complete. Advances in Applied Mathematics 3, 43–49 (1982)
9. Ribeiro, C.C., Vianna, D.S.: A hybrid genetic algorithm for the phylogeny problem using path-relinking as a progressive crossover strategy. International Transactions in Operational Research (2009)
10. Lin, Y.-M., Fang, S.-C., Thorne, J.L.: A tabu search algorithm for maximum parsimony phylogeny inference. European Journal of Operational Research 176 (2007)
11. Lin, Y.-M.: Tabu search and genetic algorithm for phylogeny inference (2008)
12. Swofford, D.L.: PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods) Version 4 (1998)
13. Hartigan, J.A.: Minimum mutation fits to a given tree. Biometrics 29 (1973)
14. Sankoff, D.: Minimal Mutation Trees of Sequences. SIAM on Applied Mathematics (1975)
15. Arvestad, L., Lagergren, J., Sennblad, B.: The gene evolution model and computing its associated probabilities. J. ACM 56, 1–44 (2009)
16. Kimura, M.: A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. Journal of Molecular Evolution (1980)
17. Guindon, S., et al.: New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. Systematic Biology (2010)
18. Guindon, S., Delsuc, F., Dufayard, J.F., Gascuel, O.: Estimating maximum likelihood phylogenies with PhyML. Methods Mol. Biol. 537, 113–137 (2009)
19. Ronquist, F., Huelsenbeck, J.P.: MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19, 1572–1574 (2003)
20. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. Journal of Classification 9, 91–116 (1992)
21. Cilibrasi, R., Vitany, P.M.B.: A New Quartet Tree Heuristic for Hierarchical Clustering. Presented at the Theory of Evolutionary Algorithms, Dagstuhl, Germany (2006)
22. Schmidt, H.A., Strimmer, K., Vingron, M., Haeseler, A.: TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing. BMC 18 (2002)
23. Snir, S., Yuster, R.: Reconstructing approximate phylogenetic trees from quartet samples. In: The 21 Annual ACM-SIAM Symposium on Discrete Algorithms, Texas (2010)
24. Tao, J., Kearney, P., Li, M.: Orchestrating quartets: approximation and data correction. In: Proceedings of 39th Annual Symposium on Foundations of Computer Science (1998)
25. G. E. M. L. E., Lupo, P.: Gene-Gene Interactions in the Folate Metabolic Pathway and the Risk of Conotruncal Heart Defects. Journal of Biomedicine and Biotechnology (2010)

26. Pereira-Leal, J., Levy, E.D., Teichmann, S.A.: The origins and evolution of functional modules: lessons from protein complexes. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 361, 507–517 (2006)
27. Lu, Y.-C., Yec, W.C., Ohashi, P.S.: LPS/TLR4 signal transduction pathway. Cytokine (2008)
28. Uetz, P., et al.: A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. Nature 403, 623–627 (2000)
29. Craig, T.: A simplified NP-complete satisfiability problem. Discrete Applied Mathematics (1984)
30. Berman, P., Alex, M.K., Scott, E.D.: Computational complexity of some restricted instances of 3-SAT. Discrete Applied Mathematics 155, 649–653 (2007)
31. Doran, R.W.: The Gray Code. Journal of Universal Computer Science 13 (2007)
32. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Series of Books in the Mathematical Sciences. W. H. Freeman (1979)
33. Day, W.H.E.: Properties of the nearest neighbor interchange metric for trees of small size. Journal of Theoretical Biology 101, 275–288 (1983)