COMPUTING THE ASSIGNMENT OF ORTHOLOGOUS GENES VIA GENOME REARRANGEMENT *

X. CHEN, J. ZHENG AND Z. FU

Department of Computer Science and Enginerring,
University of California, Riverside, CA
E-mail: xinchen, zjie, zfu@cs.ucr.edu

P. NAN AND Y. ZHONG

Shanghai Center for Bioinformatics Technology, Shanghai, China
E-mail: pnan@scbit.org, yangzhong@fudan.edu.cn

S. LONARDI

Department of Computer Science and Enginerring,
University of California, Riverside, CA
E-mail: stelo@cs.ucr.edu

T. JIANG

Department of Computer Science and Enginerring,
University of California, Riverside, CA
Shanghai Center for Bioinformatics Technology, Shanghai, China
E-mail: jiang@cs.ucr.edu

The assignment of orthologous genes between a pair of genomes is a fundamental and challenging problem in comparative genomics. Existing methods that assign orthologs based on the similarity between DNA or protein sequences may make erroneous assignments when sequence similarity does not clearly delineate the evolutionary relationship among genes of the same families. In this paper, we present a new approach to ortholog assignment that takes into account both sequence similarity and evolutionary events at genome level, where orthologous genes are assumed to correspond to each other in the most parsimonious evolving scenario under genome rearrangement. It is then formulated as a problem of computing the *signed reversal distance with duplicates* between two genomes of interest, for which an efficient heuristic algorithm was given by introducing two new optimization problems, *minimum common partition* and *maximum cycle decomposition*. Following this approach, we have implemented a high-throughput system for assigning orthologs on a genome scale, called SOAR, and tested it on both simulated data and real genome sequence data. Compared to a recent ortholog assignment method based entirely on homology search (called INPARANOID), SOAR shows a marginally better performance in terms of sensitivity on the real data set because it was able to identify several correct orthologous pairs that were missed by INPARANOID. The simulation results demonstrate that SOAR in general performs better than the iterated exemplar algorithm in terms of computing the reversal distance and assigning correct orthologs.

*This work was supported in part by NSF grants ITR-0085910, CCR-0309902, and DBI-0321756, a DOE Genomics:GTL subcontract, and National Key Project for Basic Research (973) grant 2002CB512801.

1. Introduction

Orthologs and paralogs were originally defined by Fitch[1] in 1970. Orthologs are genes in different species that evolved from the same gene in the last common ancestor of the species, and paralogs are genes that were duplicated from a single gene on the same genome. In order to avoid ambiguity, Sonnhammer and Koonin[2] further divided paralogs into two subtypes: inparalogs and outparalogs. Outparalogs between two species are paralogs in a species that were duplicated before the speciation (*i.e.,* split of the two species) and inparalogs are duplicated after the speciation. For a given set of paralogs on a genome, there commonly exists a gene that is the direct descendant of the ancestral gene of the set, namely the one that best reflects the original position of the ancestral gene in the ancestral genome. Sankoff[3] called such a gene the *true exemplar* of the paralogous set.

Orthologous genes are typically evolutionary and functional counterparts in different species. Many existing computational methods for solving various biological problems, *e.g.,* the inference of functions of new genes, the analysis of phylogenetic relationship between different species and many tools in comparative genomics, use orthologs in a critical way. As a consequence, the identification of orthologs, especially direct descendants of ancestral genes in current species, is a fundamental problem in computational biology. It follows from the definitions of ortholog and paralogs that the best way to recognize orthologs is to measure the divergence time between homologous genes in two different genomes. As the divergence time could be estimated by comparing the DNA/protein sequences of genes, most of existing algorithms for ortholog assignment, such as the well-known COG system[4,5] and INPARANOID program[6], employ a homology search algorithm such as BLAST[7]. However, the evolutionary rates of all genes in a homologous gene set may vary greatly and thus the estimation of divergence times from sequence similarity can be inaccurate. Therefore, information from homology search alone may not be sufficient for recognizing orthologs reliably, although it is usually sufficient for identifying paralogs. On the other hand, observe that molecular evolution proceeds in two different forms: local mutations and global rearrangements. Local mutations include base substitution, insertion and deletion, and global rearrangements include genome inversion, translocation, transposition, *etc*. The homology-based ortholog assignment methods only use local mutations and neglect genome rearrangement events that might actually provide valuable information.

In this paper, we propose a new approach for assigning orthologs by taking into account both local mutations and genome rearrangement events. Our method starts by identifying sets of paralogs (*i.e.,* gene families) on each genome and the correspondence between these families by using homology search. The paralogs are then treated as copies of the same genes, and ortholog assignment is formulated as a natural optimization problem of rearranging one genome consisting of a sequence of (possibly duplicated) genes into the other with the smallest number of rearrangement events. This most parsimonious rearrangement process should suggest pairs of orthologous genes in a straightforward way. To simplify the discussion (and as a first attempt), we first consider only inversion events in genome rearrangement. The above optimization problems thus becomes computing the *signed reversal distance with duplicates* (SRDD) between two genomes. SRDD is a simple extension of the well-known problem of sorting by reversals.[8] Although the problem of sorting by reversals has been intensively studied in the past decade, SRDD has basically been untouched. We give an efficient and effective heuristic algorithm for solving SRDD, using the techniques of *minimum common partition* of two given genomes and *maximum cycle decomposition* on a breakpoint graph. Based on this algorithm, we develop a high-throughput system for assigning orthologs on a genome scale, called SOAR (*i.e.,* System for Ortholog Assignment by Reversals). The heuristic algorithm for SRDD and system SOAR have been tested on both simulated and real genomic sequence data (from human, mouse and rat X chromosomes), and compared with two existing algorithms in the literature, the exemplar algorithm[3] (actually, an iterative version of it) and INPARANOID[6]. The test results demonstrate that our heuristic algorithm for SRDD in general performs better than the iterated exemplar algorithm in terms of computing the reversal distance and assigning correct orthologs, and SOAR is marginally better than INPARANOID in terms of the sensitivity of its predictions, because it was able to find several true orthologous pairs that were missed by INPARANOID. [a]

The rest of the paper is organized as follows. The next section reviews some related work on ortholog assignment and genome rearrangement. Section 3 introduces the system SOAR that we have implemented for ortholog assignment. The heuristic algorithm for SRDD, which is used in SOAR, is described in Section 4. Preliminary experiments on simulated data and real data are presented in Section 5. Some concluding remarks are given in Section 6. Due to space limitations, all proofs and some figures are omitted in the extended abstract.

2. Related Work

Since our approach for ortholog assignment combines homology search with genome rearrangement, we briefly review some related work in ortholog assignment and genome rearrangement.

2.1. *Previous results on ortholog assignment*

One of the earliest systems developed to identify orthologs is the COG database[4,5], which is widely used to infer the functions of new genes from annotated genes. COG stands for *clusters of orthologous groups*, each of which consists of individual orthologous genes or orthologous sets of paralogs from at least three lineages. When a gene is queried against the COG database, it is classified into a COG if it has at least three best BLAST hits in COG. However, as its name implies, the COG database actually does not report exactly which gene is the ortholog of the query gene if paralogs exist. That is, it does not distinguish the true ortholog of the query gene and its paralogs.

[a]SOAR is slightly worse than INPARANOID in terms of specificity. However, the specificity numbers may not be very reliable here because we validate orthologs by matching their names in Genbank. It is well known that the gene names in Genbank are not completely consistent at the present stage and orthologous genes could be named differently. Besides, the validation method is biased in favor of homology-based ortholog assignment methods since many genes in Genbank were named via homology search.

Yan *et al.*[9] proposed an approach based on the analysis of *reconciled trees*. The authors first construct a gene tree for a set of homologous genes, and then a phylogenetic (species) tree for the species from which the genes came. When there are discrepancies between these two trees, a reconciled tree is computed taking into consideration the presence of gene duplications, which results in an assignment of orthologous genes. This approach fails when the involved species belong to the same genus because the species background cannot provide sufficient information for resolving the genes under consideration.[9] Furthermore, accurate reconstruction of gene and species trees is a nontrivial problem itself, which further limits the effectiveness of the approach as a method for assigning orthologous genes. A similar approach that uses a set of *bootstrap* trees instead of reconciled trees was proposed in Storm and Sonnhammer[10] recently. These tree-based methods are not designed to assign orthologs between complete genomes.

To avoid potential errors that might be introduced by multiple alignments or phylogenetic trees, a homology-based tool called INPARANOID, was introduced in Remm *et al.*[6] It uses all-*versus*-all pairwise gene sequence comparison and *bi-directional best hits* (BBHs) to identify the so called *main orthologs* (and their inparalogs) between a given pair of genomes. This approach is in general pretty reliable; however, it could miss many pairs of orthologs or even assign orthologs incorrectly when the similarity between gene sequences does not accurately reflect the evolutionary relationship among the genes.

Recently, Cannon and Young[11] developed a suite of programs called OrthoParaMap to distinguish orthologs from paralogs, which may represent the first effort to assign orthology by making use of comparative genomic positional information. These information are retrieved from conserved *synteny blocks* in the two species of interest and then mapped onto gene family phylogeny to infer gene duplication mechanism like speciation, segmental duplications or tandem gene duplications. However, it is widely believed that genome rearrangement scenarios can essentially provide more valuable positional information than synteny blocks. This makes the assignment of orthologous genes via genome rearrangement a very promising approach.

2.2. *Genome rearrangement*

Hannenhalli and Pevzner[8] developed an elegant algorithm for computing *signed reversal distance* and *translocational distance* between two genomes (with distinct genes) in polynomial time. A permutation is represented as a *breakpoint graph*. The reversal distance between the permutation and the identity permutation is calculated by decomposing the breakpoint graph into a maximum number of edge-disjoint cycles and counting the numbers of breakpoint, cycles, and *hurdles*, as well as checking the existence of a *fortress*. The best running time for sorting a permutation by the minimum number of reversals is quadratic[12], although the signed reversal distance can computed in linear time.[13]

As mentioned before, the above model and algorithms only deal with genomes consisting of distinct genes. When studying divergent genomes that contain many highly homologous genes (*i.e.,* paralogs) scattered across each genome, we need consider genome rearrangement with *duplicated genes*. Sankoff[3] designed an *exemplar* algorithm to find the

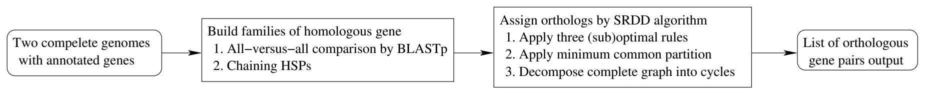

Two compelete genomes with annotated genes
Build families of homologous gene
1. All–versus–all comparison by BLASTp
2. Chaining HSPs
Assign orthologs by SRDD algorithm
1. Apply three (sub)optimal rules
2. Apply minimum common partition
3. Decompose complete graph into cycles
List of orthologous gene pairs output

Figure 1. An outline of SOAR.

genes from two corresponding gene families that are direct descendants of the same gene in the most recent common ancestral genome. The basic idea is to delete all but one member of every gene family on each of the two genomes being compared, so as to minimize some rearrangement distance over all choices of reduced genomes thus derived.

El-Mabrouk[14] recently proposed an algorithm to reconstruct an ancestral genome giving rise to the minimal number of duplication transpositions and reversals, which works only for genomes containing sets of gene families of sizes at most two. Tang and Moret[15] presented a straightforward approach by enumerating all the possible assignments of orthologs between two genomes with duplicates. However, the number of such possible assignments grows exponentially as the number of paralogs increases, making their approach applicable only to genomes with a very small number of duplicated genes.

3. A System for Ortholog Assignment Based on Sorting by Reversals

Our proposed approach for ortholog assignment takes into account both local mutations at the gene level and rearrangements at the genome level. Local mutations are measured by sequence similarity and genome rearrangement is measured by the minimum number of rearrangement events (*i.e.,* inversions of contiguous segments of genes) necessary to transform one genome into the other. Following this approach, we have implemented a system, called SOAR, that consists of two major steps: (i) identify paralogs and construct gene families from an annotated genome via homology search, and (ii) assign orthologs via a heuristic algorithm for SRDD. The steps are illustrated in Figure 1 and described separately below.

3.1. *Construction of gene families*

SOAR takes as input two annotated genomes (or chromosomes). Gene protein sequences as well as their locations are extracted to form a gene list for each genome, which is then formatted and indexed to facilitate an all-*versus*-all gene sequence comparison by BLASTp.[7] BLASTp generates several *high scoring segment pairs* (HSPs) for each pair of genes. Since the HSPs may overlap with each other, a simple summation of their scores may overestimate the similarity between the two genes. We apply a chaining algorithm[16] to find a set of compatible HSPs while maximizing their combined E-value. As in Remm *et al.*[6], two genes are considered homologous if (1) the combined E-value is less than $1e$-20 and (2) the compatible HSPs span 50% of each gene in length. Each set of homologous genes from the same genome constitute a gene family.

Since we do not attempt to consider gene loss or insertion events after the speciation in this preliminary version of SOAR, we remove from a gene family those members that are the least homologous to the members of the counterpart family so that corresponding gene families from both genomes always have equal sizes. Finally, all the remaining genes from each genome are ordered according to their locations on the genome, and the resulting two sequences of genes will be input to the following heuristic algorithm for SRDD.

3.2. *Identification of orthologs using a heuristic algorithm for SRDD*

Now we have two genomes of equal content and all the genes are assumed to be the direct descendant of their ancestral genes in the most recent common ancestral genome. The sequence of genes on each genome thus evolved from the sequence of genes on the common ancestral genome by genome rearrangement events such as reversals and transpositions. By reconstructing the most parsimonious rearrangement scenario, we could identify pairs of genes, where each pairs contains one gene from each genome, that came from the same ancestral gene and thus likely candidates for orthologs. Equivalently, we may consider the most parsimonious transformation from one genome into the other by reversals and transpositions. Since transpositions occur much less frequently than reversals, hereafter we will only consider reversals. The following is a simple example to illustrate the idea of ortholog assignment via sorting by reversals. Consider two genomes, $G = +c - b + a + b$ and $H = +a - b - b - c$, consisting of four genes and one multi-gene family each. The most parsimonious transformation from G into H uses only three reversals. In this transformation, the first (or second) copy of gene b in G is found to correspond to the first (or second, respectively) gene b in H, indicating that they might be a pair of orthologous genes.

This approach for ortholog assignment raises a new computational problem, *i.e.,* how to sort a sequence of genes (with duplicates) into another with the minimum number of reversals (SRDD). SOAR uses an efficient and effective heuristic algorithm for SRDD, which will be explained in the next section.

4. An Efficient Heuristic Algorithm for SRDD

In this section, a genome is represented as a string of signed symbols from a finite alphabet $\mathcal{A}$, where each sign (+ or -) represents a transcriptional orientation and a symbol denotes a gene. All occurrences of a symbol in a genome constitute a gene family. A gene is called a *singleton* if it is the only member of its family; otherwise, it is a *duplicated* gene. Two genomes G and H are *related* if they have the same gene content, Given two related genomes G and H, the *reversal distance problem* is to find the smallest number of reversals $\rho_1, \rho_2, \cdots, \rho_t$ such that $G \cdot \rho_1 \cdot \rho_1 \cdots \rho_t = H$. The *reversal distance* between G and H is thus $d(G, H) = t$. If all the genes in G and H are singletons, the reversal distance problem is usually referred to as the problem of *sorting by reversal*, and the distance can be calculated by the Hannenhalli-Pevzner (H-P) formula[8]:

$$d(G, H) = b(G, H) - c(G, H) + h(G, H) + f(G, H)$$

where $b(G, H)$ is the number of black edges in the breakpoint graph for G and H, $c(G, H)$ the number of cycles in maximum cycle decomposition, $h(G, H)$ the number of hurdles, and $f(G,H)$ the number of fortresses, respectively. [b]

When G and H contain duplicated genes, however, the problem cannot be directly solved by the H-P algorithm anymore. Once ortholog assignment between the two genomes is accomplished, the signed reversal distance with duplicates (SRDD) can be simply computed using the H-P formula since every gene can then be regarded as unique. Let us denote by M the set of all the possible ortholog assignment. Given an assignment m and a genome G, let G^m denote gene sequences of G after orthologs have been assigned by m. The following straightforward lemma relates SRDD to the assignment of orthologs.

Lemma 4.1. *Given genomes G and H, we have* $d(G,H) = min_{m\in M} d(G^m, H^m)$.

Unfortunately, the following theorem shows that SRDD is NP-hard. Observe that the NP-hardness of sorting unsigned strings over a fixed alphabet[17] does not imply Theorem 4.2.

Theorem 4.2. *SRDD is NP-hard, even when the maximum size of a gene family is limited to two.*

In the following, we devise an efficient and effective heuristic algorithm for SRDD. To simplify the discussion, we will assume without loss of generality that the first genes and the last genes of the two related genomes are identical and are positive singletons.

4.1. *A useful lower bound*

The following definitions are adapted from Christie and Irving[17] and El-Mabrouk[14]. We convert a (signed) genome $G = (g_1 g_2 \cdots g_n)$ to an unsigned one by replacing each gene g_i with a string $g_i^h g_i^t$ if g_i is positive or $g_i^t g_i^h$ if g_i is negative, as it is done in the breakpoint graph of H-P. A *partial graph*[14] associated with a genome $G = (g_1 g_2 \cdots g_n)$ is the graph $\mathcal{G}(V,E)$, where $V = \{g_i^s | 1 \le i \le n, s \in \{h,t\}\}$, and each of (undirected) edge in E links two nodes in V that correspond to adjacent symbols in the genome G except pairs of the form g_i^h and g_i^t from the same gene g_i. Let $\tilde{V}$ be the set of distinct symbols in V, where g_i^h and g_j^h are considered as the same symbol if g_i and g_j are from the same family. Clearly, the partial graphs of a pair of related genomes have an identical vertex set V and set $\tilde{V}$. For each pair of elements $\{\tilde{v}_1, \tilde{v}_2\} \in \tilde{V}$, let $f_G(\tilde{v}_1, \tilde{v}_2)$ denote the number of edges in E that links two nodes in the partial graph $\mathcal{G}(V,E)$ of G with symbols $\tilde{v}_1$ and $\tilde{v}_2$, respectively. The *number of reversal breakpoints* between two related genomes G and H is defined as follows[17]

$$b_r(G,H) = \sum_{\{\tilde{v}_1,\tilde{v}_2\}\in\tilde{V}} \delta(f_H(\tilde{v}_1,\tilde{v}_2) - f_G(\tilde{v}_1,\tilde{v}_2))$$

where $\delta(x) = x$ if $x > 0$ and 0 otherwise. The following theorem follows from the observation that a reversal operation could reduce the number of breakpoints by at most two.

[b]Please refer to Hannenhalli and Pevzner[8] for the definitions of black edges, hurdles, and fortresses.

Theorem 4.3. *Let G and H be a pair of related genomes. Their reversal distance is lower bounded by $d(G,H) \geq \lceil b_r(G,H)/2 \rceil$.*

4.2. *(Sub)Optimal assignments*

Our heuristic algorithm begins by finding individual ortholog assignments that are (nearly) optimal with respect to SRDD. Note that the (correct) identification of each ortholog reduces the number of duplicates in the related genomes and thus makes the SRDD problem easier. Lemma 4.4 gives a nearly optimal rule, whereas Lemma 4.5 gives an optimal assignment rule.

Lemma 4.4. *Assume that $g_{i-1}g_ig_{i+1}$ is identical to $h_{j-1}h_jh_{j+1}$ or its reversal, where g_{i-1} and g_{i+1} are singletons but g_i is not. Define two new genomes $G^{'}$ and $H^{'}$ from G and H by assigning orthology between g_i and h_j. Then, $d(G,H) \leq d(G^{'},H^{'}) \leq d(G,H)+1$.*

Lemma 4.5. *Assume that $g_{i-1}g_i$ and $g_{j-1}g_j$ are identical to $h_{k-1}h_k$ and $h_{l-1}h_l$ or their reversals, respectively. Suppose that each of the above four pairs is composed of a singleton and a duplicated gene, and the four duplicated genes are from the same family and without any other gene included. Define two new genomes $G^{'}$ and $H^{'}$ from G and H by assigning orthology between the duplicated genes in $g_{i-1}g_i$ and $h_{k-1}h_k$ and between the duplicated genes in $g_{j-1}g_j$ and $h_{l-1}h_l$. Then, $d(G,H) = d(G^{'},H^{'})$.*

It is well known that in the problem of sorting signed permutations by reversals, the distance value is highly dominated by the first two terms of the H-P formula, *i.e.,* $\tilde{d}(G,H) = b(G,H) - c(G,H)$. Based on this observation, one can devise the following assignment rule, which unfortunately does not guarantee optimality with respect to SRDD.

Lemma 4.6. *Assume that $g_{i-1}g_i$ is identical to $h_{j-1}h_j$ or its reversal, where each of pair consists of a singleton and a duplicated gene. Define two new genomes $G^{'}$ and $H^{'}$ from G and H by assigning orthology between the duplicated genes in $g_{i-1}g_i$ and $h_{j-1}h_j$. Then, $\tilde{d}(G,H) = \tilde{d}(G^{'},H^{'})$.*

4.3. *Minimum common partition*

In order to formulate an efficient algorithm for the problem of SRDD, we introduce here for the first time a new optimization problem, called *minimum common partition* (MCP). Let us first define a *segment* $\bar{g}_i$ as a substring (or its reversal) of a genome sequence G. A *partition* is a list $\{\bar{g}_1, \bar{g}_2, \cdots, \bar{g}_n\}$ of segments of a genome G, such that the concatenation of the segments (or their reversals) in some order results in the genome G. The list can be thought as a *contracted representation* of G if we consider each segment $\bar{g}_i$ as a symbol. A list of segments is called a *common partition* of two related genomes G and H if it is a partition of G and a partition of H as well. Note that the contracted representations of two related genomes induced by a common partition are still related to each other. Furthermore, a *minimum common partition* is a partition with the minimum cardinality (denoted as $L(G,H)$) over all choices of common partitions of G and H. For example, for genomes

$G = +c + a + b - a + d$ and $H = +c + a - b - a + d$, a minimum common partition is $\{+c+a, +b, -a+d\}$ with $L(G, H) = 3$. The *minimum common partition* (MCP) problem is naturally defined as the problem of finding the minimum common partition between two given genomes. The following theorem establishes the relation between SRDD and MCP.

Theorem 4.7. *Given two related genomes G and H, we have $\lceil (L(G, H) - 1)/2 \rceil \leq d(G, H) \leq L(G, H) - 1$.*

Theorem 4.7 suggests a way to approximate SRDD by MCP. Unfortunately, MCP is also NP-hard.[18]

Theorem 4.8. *Let k-MCP denote the version of MCP where each gene family is of size at most k. The problem k-MCP is NP-hard, for any $k \geq 2$.*

Now we present an approximation algorithm for MCP. Given two related genomes G and H, a *single-match* is a pair of identical genes g_i and h_j from G and H that may have different signs. A *pair-match* is a pair of adjacent gene pairs $g_i g_{i+1}$ and $h_j h_{j+1}$ that are identical or the reversal of each other. Clearly, a pair-match consists of two single-matches. A common partition $\{\bar{g}_1, \bar{g}_2, \cdots, \bar{g}_n\}$ between G and H can be considered as a one-to-one mapping M from G to H. When $M(g_i) = h_j$, g_i and h_j form a single-match.

Observe that two pair-matches may not co-exist in a common partition. Such pair-matches are said to be *incompatible*. For two related genomes G and H, we can construct a *pair-match* graph $\mathcal{P}(V, E)$, where V consists of all possible pair-matches between G and H, and E includes edges connecting incompatible pair-matches. The following lemma is straightforward.

Lemma 4.9. *The maximum independent set problem on $\mathcal{P}(V, E)$ is equivalent to the minimum common partition problem.*

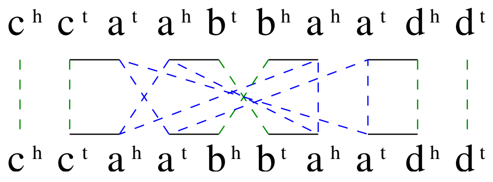
Since the complement of an independent set of $\mathcal{P}(V, E)$ is a vertex cover of $\mathcal{P}(V, E)$, we can approximate MCP by using an efficient approximation algorithm for the vertex cover (VC) (*e.g.,* the standard greedy algorithm with approximation ratio 2). An outline of our approximation algorithm is shown in Figure 3.

If one assumes that the approximation algorithm for vertex cover has ratio r, we can obtain an upper bound on the performance of APPROX-MCP as follows.

Lemma 4.10. *If the size of the common partition found by the* APPROX-MCP *algorithm is l, then $l \leq (r - 1)(|V| - n) + r \cdot L(G, H)$, where $|V|$ is the size of the vertex set of the pair-match graph and n is the size of genome G. In particular for 2-MCP, the above algorithm achieves an approximation ratio of 1.5.*

4.4. *Maximum cycle decomposition*

The contracted representations of the genomes G and H may still contain duplicates, although we do not expect the number of duplicates to be large. Again, we define another



c h c t a t a h b t b h a h a t d h d t
c h c t a h a t b h b t a h a t d h d t

Figure 2. The complete graph of two related genomes $G = +c - a - b + a + d$ and $H = +c + a + b + a + d$. Each gene is represented by two vertices in the graph.

new problem, called *maximum cycle decomposition* (MCD), to complete the solution for SRDD.

The *complete graph* associated with a pair of related genomes $G = (g_1 g_2 \cdots g_n)$ and $H = (h_1 h_2 \cdots h_n)$, denoted $\mathcal{G}(V, E)$, includes the partial graphs of both G and H as subgraphs, plus cross-genome edges joining g_i^s and h_j^s, where $1 \leq i \leq n$, $1 \leq j \leq n$, $s \in \{t, h\}$ if g_i and h_j are identical genes. For example, the complete graph for genomes $G = +c - a - b + a + d$ and $H = +c + a + b + a + d$ is shown in Figure 2.

The *maximum cycle decomposition* (MCD) problem is the problem of decomposing a given complete graph into a maximal set of cycles such that (1) every vertex belongs to exactly one cycle, except for the first and last vertices of each genome; (2) the two vertices representing each gene must be connected respectively to the two vertices of some (identical) gene in the other genome by edges of the cycles, *i.e.,* the connections satisfy a pairing condition; and (3) edges within a genome and across genomes alternate in a cycle. The following theorem is a simple extension of the H-P formula and gives tighter bounds on reversal distance in terms of MCD compared to the bounds in terms of MCP in the previous section.

Theorem 4.11. *Given two related genomes G and H, we have* $n - 1 - C \leq d(G, H) \leq n - 1 - C_4$*, where* n *is the size of* G*,* C *is the number of cycles in the maximum cycle decomposition, and* C_4 *is the maximal number of cycles of size four in any feasible solution to MCD.*

We do not know the complexity of MCD, but we believe it is NP-hard because the problem of decomposing a general graph into a maximum number of vertex-disjoint cycles is NP-hard.[19] We use a greedy algorithm in SOAR to solve MCD, as outlined in Figure 3. The basic idea of the algorithm is to find small cycles that satisfy the above three conditions. Intuitively, small cycles result in large cycle decompositions, although it is not always the case.

Our heuristic algorithm for SRDD combines the three (sub)optimal ortholog assignment rules, APPROX-MCP and GREEDY-MCD, as outlined in Figure 3. We note that this algorithm actually works for genomes with unequal gene contents, although the above discussion assumes related genomes with equal gene content.

```
Algorithm APPROX-MCP($G$, $H$)
/* $G$ and $H$ are a pair of related genomes. */
1. Construct the pair-match graph $\mathcal{P}(V, E)$ for $G$ and $H$
2. Find an approximation of the vertex cover $C$ of $\mathcal{P}$
3. Identify the segments based on the pair-matches in $V - C$
4. Output all the segments as a common partition of $G$ and $H$

Algorithm GREEDY-MCD($G$, $H$)
1. Construct the complete graph $\mathcal{G}(V, E)$ for $G$ and $H$
2. while $V$ is not empty do
   a. Select a vertex from $V$
   b. Find a shortest cycle that passes through the selected
      vertex while not violating the pairing constraint
   c. Remove all vertices and edges in the shortest cycle
3. Output all the cycles found as a cycle decomposition

Algorithm ALG-SRDD($G$, $H$)
1. Apply the three (sub)optimal rules given in Lemmas 4.4 – 4.6
2. Run the APPROX-MCP algorithm
3. Run the GREEDY-MCD algorithm
4. Sort $G$ into $H$ according to the above cycle decomposition
```

Figure 3. An efficient approximation algorithm for MCP, a greedy algorithm for maximum cycle decomposition, and an outline of the heuristic for SRDD.

5. Experimental Results

In order to test the performance of SOAR as a tool to assign orthologs, we have applied it to both simulated data and real genome sequence data, and compared its results with two algorithms in the literature, namely, an iterated version of the exemplar algorithm[3] and INPARANOID[6].

5.1. *Simulated data*

We use simulated data to assess the performance of our heuristic algorithm for SRDD. In order to make a comparison test, we implemented the exemplar algorithm of Sankoff[3] and extended it into a tool for assigning orthologs. Although the exemplar algorithm was not originally proposed for the purpose of ortholog assignment, its objective is closely related to solving the ortholog assignment problem. Namely, it looks for a pair of homologous (or identical) genes from two genomes that are direct descendants of the same gene in their most recent common ancestral genome. Therefore, for any two given genomes, the pairs of genes found by the exemplar algorithm should be orthologous to each other. We can iterate the exemplar algorithm by making the output orthologs singletons and repeating the algorithm on the new genomes again and again, until no duplicated genes are left in the genomes.

The simulated data is generated as follows. Start from a genome G with n distinct symbols whose signs is generated randomly. Each symbol defines a single gene family.

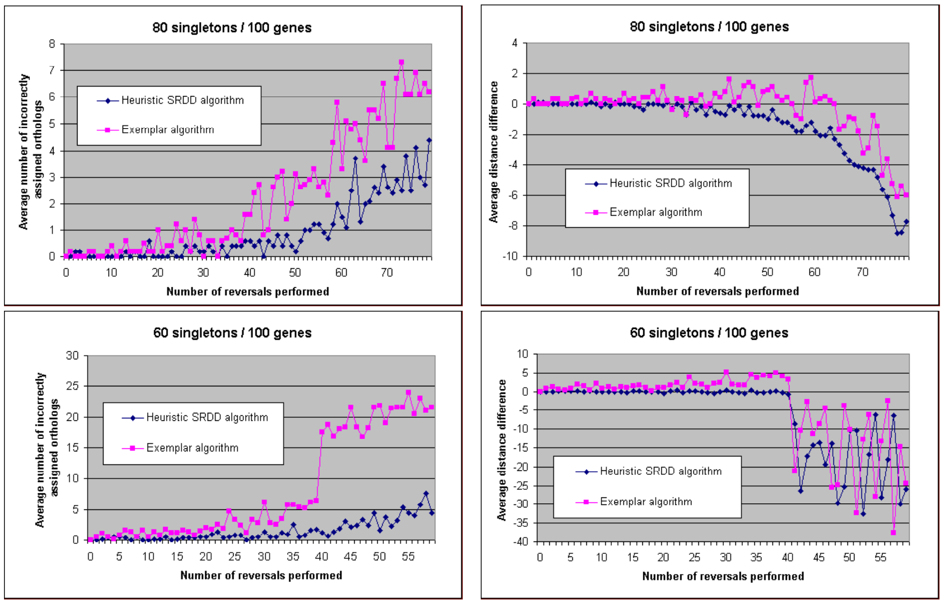


Figure 4. Comparison of ALG-SRDD and the exemplar algorithm on simulated data.

Then, randomly combine two gene families into a new family until r singletons are left in the genome G. In order to obtain a related genome H, perform k reversals on the genome G. The boundaries of these reversals are uniformly distributed within the size of the genome. The triple (n, r, k) specifies the parameters for generating a pair of related genomes.

We ran the exemplar algorithm and ALG-SRDD on 10 random instances for each combination of parameters. Figure 4 shows the average performance of both algorithms over 10 instances in terms of the number of incorrectly assigned orthologs and the reversal distance. On the average, the number of genes with incorrectly assigned orthologs generally increases as the number of reversals k increases. However, our heuristic always produces fewer incorrect ortholog assignments than the exemplar algorithm. In fact, the gap grows with the number of reversals and the number of duplicates. For $n = 100$ and $r = 80$, our algorithm on average produces at most one mistake for up to 50 reversals and at most 5 mistakes for up to 80 reversals. These statistics indicate that our method is quite reliable in assigning orthologs.

5.2. *Real genome sequence data*

We use the X chromosomes of human (*Homo sapiens*), mouse (*Mus musculus*) and rat (*Rattus norvegicus*) genomes in our real data test. Excluding genes whose (protein) sequences are not available in Genbank, we downloaded 922 genes from the human X chromosome,

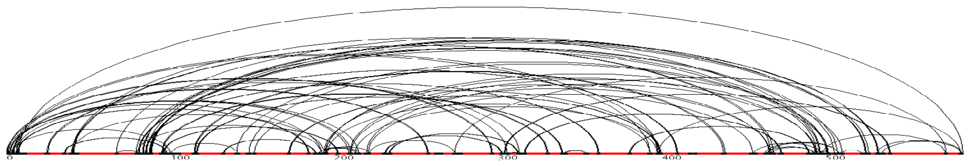


Figure 5. The breakpoint graph of human and mouse X chromosomes. Note that the pictures only show the orderings of the genes in the genomes, not their actual locations.

1030 genes from mouse, and 899 genes from rat, respectively. At the first step of SOAR, an all-*versus*-all sequence comparison was performed between each pair of genomes using BLASTp. Based on this, the families of homologous genes were constructed. Then, by using ALG-SRDD SOAR assigned 583 pairs of orthologous genes between human and mouse, 518 pairs between human and rat, and 599 pairs between mouse and rat as shown in Table 1.

The genome rearrangement between human and mouse X chomosomes that ALG-SRDD found consists of 123 reversal operations, and its breakpoint graph is depicted in Figure 5, which includes 143 breakpoints. In contrast, by studying synteny blocks instead of genes in the genomes, Pevzner and Tesler[20] report that there are at least 7 macrorearrangements (with a DNA sequence span $>$ 1 Mb) and 177 microrearrangements, which may be inaccurate due to assembly errors. Note that the synteny blocks are required to be unique when generated so that the H-P formula can be applied to calculate the reversal distance. The breakpoint graphs showing genome rearrangement for human/rat (117 reversals and 135 breakpoints) and for mouse/rat (155 reversals and 188 breakpoints), both produced by ALG-SRDD, can be found in at the SOAR webpage (`http://www.cs.ucr.edu/~xinchen/soar.htm`). With these breakpoint graphs orthologs between two genomes are assigned in a straightforward way, and the resulting dot-plot graph showing ortholog mappings between human and mouse X chromosomes is illustrated in Figure 6. Although this dot-plot graph is very similar to the one obtained by Pevzner and Tesler[20] using (unique) synteny blocks, an interesting and noticeable difference between them occurs in the synteny block spanning from 78,691,856bp to 83,712,889bp in human X chromosome with the block spanning from 100,126,658bp to 105,951,929bp in mouse X chromosome. Pevzner and Tesler[20] reported that these two synteny blocks correspond to each other with opposite orientations. However, for all the six gene pairs that are located in the two blocks and assigned orthology by our SOAR system, each has the same transcription direction, which instead suggests the same orientation of the whole blocks. We suspect that the anchors (*i.e.*, BBHs) used in Pevzner and Tesler[20] to define these synteny blocks are all from noncoding regions.

Unfortunately, not much is known (or has been experimentally verified) about the true orthologs among these genes because the functions of the genes are mostly unknown. Therefore, we took an indirect approach to validate our assignments by using the gene annotation information in Genbank, in particular, gene names (or symbols). The name of a gene is usually given to convey the character or function of the gene.[21] Genes with identi-

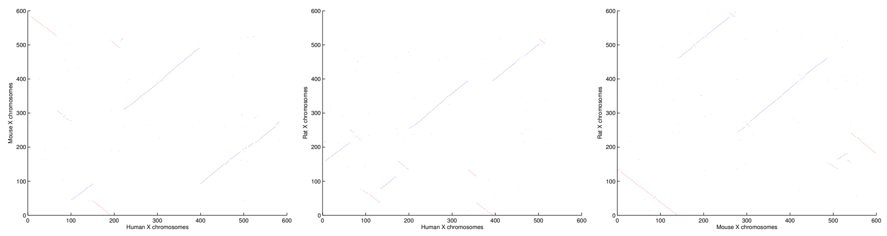


Figure 6. Dot-plot graphs showing ortholog mapping between pairs of human, mouse and rat X chromosomes. The X-axis/Y-axis consists of genes from human/mouse/rat X chromosome. A blue point in the graph represents an orthologous pair of two genes having the same transcriptional direction, while a red point represents an orthologous pair of two genes that have different transcriptional directions.

cal names are likely to be an orthologous pair, although genes with different names could still be orthologs due to naming inconsistency in Genbank. We extracted the names of the above genes from Genbank. Some of the names begin with "LOC", implying that these genes have not yet been assigned official names/symbols. Gene pairs involving such names are ignored in all subsequent statistics. If a pair of genes output by SOAR have completely identical names, we count them as a true positive pair; otherwise it is a false positive pair. We also calculated the total number of *assignable* pairs of orthologs between each pair of X chromosomes, *i.e.,* the total number of pairs of genes with identical names. For example, there are 300 assignable ortholog pairs between human and mouse. Among the 583 ortholog pairs predicted by SOAR, 284 are true positives, 128 involve genes without official names and 171 are false positives, resulting in a sensitivity of 94.6% and a specificity of 62.4% (See Table 1 for definitions). Observe that the specificity number may not be accurate because genes with different names could still be orthologs.

In order to compare SOAR with existing homology-based methods for ortholog assignment, we implemented the INPARANOID algorithm described in Remm *et al.*[6] The latter algorithm relies on BBHs (*i.e.,* bi-directional best hits) between genes from two genomes. In practice, BBHs have been widely used to assign orthologs between two species, *e.g.,* the HomoloGene database of Genbank (`http://www.ncbi.nlm.nih.gov/HomoloGene`). The comparative results on all three pairs of genomes are summarized in Table 1. The results demonstrate that SOAR and INPARANOID in general make similar assignments, although they use completely different methods to assign orthologs, *i.e.,* one is entirely based on homology search (*e.g.,* BBH) and the other relies mainly on genome rearrangement. SOAR performs slightly better than INPARANOID in terms of sensitivity, even though the validation method is a bit biased in favor of homology-based ortholog assignment methods since many genes in Genbank were named via homology search.

6. Conclusion and Future Research

In this paper, we presented a novel approach to ortholog assignment that takes into account both sequence similarity and evolutionary events (*i.e.,* reversals) at the genome level.

Table 1. Comparison of ortholog assignments by INPARANOID and SOAR. [†]The total number of assignable ortholog pairs between two chromosomes. [‡]The total number of ortholog pairs assigned. [η]The number of true positives divided by the total number of true positives and false positives (excluding pairs with unknown names). [ς]The percentage of true positives among all assignable ortholog pairs. [ξ]The number of orthologs assigned by both INPARANOID and SOAR.

	assignable	INPARANOID			SOAR			common
	orthologs[†]	assigned[‡]	speci.[η]	sensi.[ς]	assigned	speci.	sensi.	orthologs[ξ]
human/mouse	300	527	65.5%	92.6%	583	62.4%	94.6%	509
human/rat	116	468	78.8%	93.1%	518	75.8%	92.2%	448
mouse/rat	115	542	81.8%	98.2%	599	81.4%	99.1%	524

We formulated the problem as that of computing the signed reversal distance with duplicates between the two genomes of interest. The problem was decomposed into two new optimization problems (MCP and MCD), for which we designed and analyzed efficient algorithms. We implemented the algorithm in a system for assigning orthologs on a genome scale, called SOAR.

Our preliminary experiments on simulated and real data have demonstrated that ortholog assignment via genome rearrangement is a very promising method. The current version of SOAR does not consider genome rearrangement events such as transposition, gene loss, and gene insertion. It also ignores the issue of inparalogs and works only with single-chromosomal genomes. We plan to look into these extensions in the future.

References

1. W.M. Fitch. Distinguishing homologous from analogous proteins. *Syst. Zool.* 19, pp.99-113, 1970.
2. E. Sonnhammer and E. Koonin. Orthology, paralogy and proposed classification for paralog subtypes. *Trends Genet.* 16, pp.227-231, 2000.
3. D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11): 909-917, 1999.
4. R.L. Tatusov, E.V. Koonin and D.J. Lipman. A genomic perspective on protein families. *Science* 278: 631-637.
5. R.L. Tatusov, M.Y. Galperin, D.A. Natale and E.V. Koonin. The COG database: A tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Res.* 28: 33-36, 2000.
6. M. Remm, C. Storm and E. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J. Mol. Biol.*, 314, pp.1041-1052, 2001.
7. S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller and D. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17): 3389-3402, 1997.
8. S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip (Polynomial algorithm for sorting signed permutations by reversals). *Proc. 27th Annual ACM Symposium on the Theory of Computing*. pp.178-187, 1995.
9. Y.P. Yuan, O. Eulenstein, M. Vingron and P. Bork. Towards detection of orthologues in sequence databases. *Bioinformatics*, 14(3), pp.285-289, 1998.
10. C. Storm and E. Sonnhammer. Automated ortholog inference from phylogenetic trees and calculation of orthology reliability. *Bioinformatics*, 18(1), 2002.
11. S.B. Cannon and N.D. Young. OrtholParaMap: distinguishing orthologs from paralogs by integrating comparative genome data and gene phylogenies. *BMC Bioinformatics*, 4(1):35, 2003.
12. H. Kaplan, R. Shamir and R. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algo-*

rithms, pp. 344-351, 1997.
13. D. Bader, B. Moret and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5): 483-491, 2001.
14. N. El-Mabrouk. Reconstructing an ancestral genome using minimum segments duplications and reversals. *Journal of Computer and System Sciences*, 65, pp.442-464, 2002.
15. J. Tang and B. Moret. Phylogenetic reconstruction from gene rearrangement data with unequal gene contents. *Proc. 8th Workshop on Algorithms and Data Structures* (WADS'03), 37-46, 2003.
16. K. Chao and W. Miller. Linear space algorithms that build local alignments from fragments. *Algorithmica*, 13: 106-134, 1995.
17. D. Christie and R. Irving. Sorting strings by reversals and by transpositions. *SIAM J. Discrete Math.* Vol. 14(2). pp.193-206, 2001.
18. A. Goldstein, P. Kolman and J. Zheng. Minimum common string partition problem: hardness and approximations. *Accepted by ISAAC'04*, 2004.
19. A. Caprara. Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM J. Discrete Math.* Vol. 12(1), pp.91-110, 1999.
20. P. Pevzner and G. Tesler. Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. *Genome Research*, 13: 37-45, 2003.
21. H.M. Wain, *et al.*. Guidelines for human gene nomenclature. *Genomics*, 79(4), pp.464-470, 2002.