ACCURATE IDENTIFICATION OF ORTHOLOG GROUPS AMONG MULTIPLE GENOMES

Guanqun Shi*, Meng-Chih Peng and Tao Jiang
Department of Computer Science, University of California, Riverside, CA 92521, USA
*Email: {*gshi, mpeng, jiang}@cs.ucr.edu*

The identification of orthologous genes shared by multiple genomes plays an important role in evolutionary studies and gene functional analyses. Based on a recently developed accurate tool, called MSOAR 2.0, for ortholog assignment between a pair of closely related genomes based on genome rearrangement, we present a new system MultiMSOAR 2.0, to identify ortholog groups among multiple genomes in this paper. In the system, we construct gene families for all the genomes using sequence similarity search and clustering, run MSOAR 2.0 for all pairs of genomes to obtain the pairwise orthology relationship, and partition each gene family into a set of disjoint sets of orthologous genes (called *super ortholog groups* or *SOGs*) such that each SOG contains at most one gene from each genome. For each such SOG, we label the leaves of the species tree using 1 or 0 to indicate if the SOG contains a gene from the corresponding species or not. The resulting tree is called a *tree of ortholog groups* (or *TOGs*). We then label the internal nodes of each TOG based on the parsimony principle and some biological constraints. Ortholog groups are finally identified from each fully labeled TOG. In comparison with a popular tool MultiParanoid on simulated data, MultiMSOAR 2.0 shows significantly higher prediction accuracy. It also outperforms MultiParanoid and the Roundup multi-ortholog repository in real data experiments using gene symbols as a validation tool. In addition to ortholog group identification, MultiMSOAR 2.0 also provides information about gene births, duplications and losses in evolution, which may be of independent biological interest.

1. INTRODUCTION

The ever-increasing number of completely sequenced genomes brings great opportunities as well as challenges to the study of comparative genomics. It makes the study of the evolutionary history of closely related species at the genome level possible. It also enhances our ability to perform gene functional analyses across different species. For these purposes as well as many other applications, the identification of orthologous genes across different species often serves as a starting point.

1.1. Definitions

Orthologous genes (*i.e., orthologs*) are genes in different genomes that evolved from a common ancestral gene through speciation events[1]. They are more likely to preserve the original gene function. As a result, orthologs are often used as universal and unique landmarks within each genome as well as links across different genomes[2].

Orthology between two genomes is usually thought of as a many-to-many relationship due to post-speciation gene duplications[3]. However, if we know which genes are the direct descendants of the ancestral genes and which are duplicated after the speciation, then we can define a one-to-one orthology relationship between the two direct descendant genes of each ancestral gene (such a pair of genes are said to form an *ortholog pair*), while treating the duplicated genes as inparalogs[4, 5].

When multiple genomes are being compared, the orthology relationship is more complicated because of the interleaving between speciation and gene duplication events. In this paper, we extend the above one-to-one orthology relationship between a pair of genomes to multiple genomes in a straightforward way and define an *ortholog group* for a given set of genomes as a maximal set of genes (from different genomes) that are the direct descendants of the same ancestral gene. Note that the genes in such an ortholog group are not separated by any gene duplication. Hence, this definition, although a bit stringent, is faithful to the original definition of orthology in Ref. 1. For example, according to this definition, there are 4 ortholog groups in Figure 1(b): $(\alpha_{4,1}, \alpha_{5,1}, \alpha_{7,1})$, $(\alpha_{4,2}, \alpha_{5,2})$, $(\alpha_{4,3})$, $(\beta_{6,1}, \beta_{7,1})$. We note in passing that other more general definitions of

*Corresponding author.

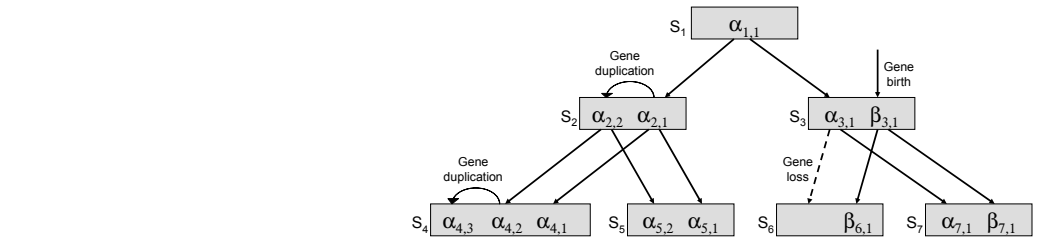
ortholog groups have been considered in the literature and used in popular orthology databases such as COG[6] and EnsemblCompara[3]. In these definitions, orthology is considered as a many-to-many relationship and thus paralogs (*i.e.*, genes that are separated by duplications) are often allowed in an ortholog group. We prefer treating orthology as a one-to-one relationship because it makes the presentation of the paper simpler and validation of our results cleaner.

1.2. Existing Ortholog Assignment Tools

Most of the traditional ortholog identification methods are based on sequence similarity search, such as COG/KOG[6], OrthoMCL[7], InParanoid/MultiParanoid[4, 8] and HomoloGene[9]. Generally speaking, these methods first calculate some pairwise similarity scores and then use some clustering algorithms to identify ortholog pairs or groups. Take the InParanoid program for example. It assigns a gene pair with the bidirectional best hit (*i.e., BBH*) as a main ortholog pair and uses it as the "seed" to cluster similar genes from both genomes into an ortholog group. As its extension to multiple genomes, the MultiParanoid program basically clusters the pairwise orthology results of InParanoid to generate ortholog groups for multiple genomes. Though the BBH requirement for a main ortholog pair seems to be reasonable when comparing two genomes, it becomes too stringent when comparing multiple genomes. As a result, the MultiParanoid program may miss a lot of true ortholog groups when some of the ortholog pairs are not BBHs. OrthoMCL is an ortholog assignment program similar to InParanoid, but uses a different clustering algorithm (the Markov Clustering algorithm, or *MCL*) to find ortholog groups for multiple genomes. However, it cannot resolve the many-to-many orthology relationship among multiple genomes effectively. As a result, the ortholog groups found by OrthoMCL may include lots of "recent" inparalogs from each genome[7].

Another popular method to identify orthologs is based on phylogenetic trees, such as TreeFam[10], PhyOP[11], and EnsemblCompara GeneTrees[3]. A phylogeny can be used conveniently to represent the evolution of a gene family. However, tree-based methods generally present orthology as a many-to-many relationship. Most of them can never tell the "parent-daughter" relationships among duplicated genes[12]. As a result, most tree-based methods cannot differentiate orthologs that are direct descendants of an ancestral gene and those inparalogs that are products of recent duplications. Consequently, each ortholog group found by these methods tends to include lots of lineage-specific duplicated inparalogs.

By taking other information into consideration, such as gene positions and genome rearrangement, some combinatorial approaches have been proposed in recent years. CCCPart is a synteny-based approach to find orthologs based on the assumption that isofunctional genes are well preserved both in common gene neighborhood as well as in sequence similarity between two or more species[13, 14]. However, it is known that genome rearrangement is very common between closely related genomes[15–18]. In fact, there might be many microrearrangments even within the same synteny block[17]. Based on genome rearrangement, a high-throughput ortholog assignment system called MSOAR[19] has been developed. It is based on the assumption that orthologs should correspond to each other on the evolutionary path that minimizes the number of rearrangements and post-speciation duplications. By dealing with tandem gene duplications explicitly using a phylogenetic approach, an improved system MSOAR 2.0 was recently reported in Ref. 5, which has been shown to outperform the original system MSOAR in terms of prediction accuracy. However, MSOAR and MSOAR 2.0 can only assign orthologs between two genomes. As an extension to MSOAR, MultiMSOAR tries to assign orthologs among multiple genomes by using a simple clustering method based on the pairwise results of MSOAR[20]. However, the MultiMSOAR program can actually handle only three genomes well. When more genomes are involved, MultiMSOAR may not find ortholog groups accurately because it does not take into account the phylogenetic relationship among the genomes. Furthermore, MultiMSOAR only considers those ortholog clusters that do not have gene losses in any species to be ortholog groups. This constraint might be acceptable for three closely related species, but it is too stringent when considering more species, since we expect



(a) (b)

(c) (d) (e) (f) (g)

Fig. 1. (a) The species tree for four species: S_4, S_5, S_6, S_7. (b) An example of genome evolution for the four species in (a). (c) The TOG for genes $\alpha_{4,1}, \alpha_{5,1}, \alpha_{7,1}$ in (b). (d) The TOG for genes $\alpha_{4,2}, \alpha_{5,2}$ in (b). (e) The TOG for gene $\alpha_{4,3}$ in (b). Note that, in this paper, we will only be interested in ortholog groups containing at least two genes, and singleton ortholog groups will be ignored since they consist of only inparalogs from individual genomes. (f) The TOG for genes $\beta_{6,1}, \beta_{7,1}$ in (b). (g) An example of a TOG labeling. The labeling suggests two ortholog groups in the TOG, one consisting of two genes from the two leftmost species and the other two genes from the last three species.

to see many gene births and losses as well as duplications in the evolutionary history. As a consequence, we should allow gene losses within an ortholog group and ortholog groups to be composed of genes from a subset of the genomes.

1.3. Current Work

In this paper, we develop a system called MultiMSOAR 2.0 to identify ortholog groups for multiple genomes. In addition to being an extension of MSOAR 2.0 to multiple genomes, MultiMSOAR 2.0 presents a new combinatorial approach for constructing ortholog groups. Compared with MultiMSOAR, MultiMSOAR 2.0 allows gene losses within an ortholog group and ortholog groups involving genes only from a subset of the genomes. It also attempts to minimize the number of gene births, losses and duplications within a gene family when assigning ortholog groups. Moreover, compared with many other ortholog assignment tools for multiple genomes, MultiMSOAR 2.0 can provide more information about genome evolution in terms of gene births, losses as well as duplications.

An outline of MultiMSOAR 2.0 is shown in Figure 2. In short, MultiMSOAR 2.0 constructs gene families for all the genomes first by using sequence similarity search (*i.e.*, BLASTp) and the clustering algorithm MCL as done in Ref. [5]. Then it applies MSOAR 2.0 to find ortholog pairs between all pairs of genomes. After that, it builds a weighted multipartite graph using the pairwise orthology information and sequence similarity between each pair of orthologs and attempts to find a maximum weight matching for each gene family. Then it partitions each family into a set of disjoint sets of orthologous genes (called *super ortholog groups* or *SOGs*) such that each SOG contains at most one gene from each genome. Each such SOG may potentially consist of several ortholog groups. In order to partition a SOG into ortholog groups, MultiMSOAR 2.0 labels the leaves of the species tree using 1 or 0 to indicate if the SOG contains a gene from the corresponding species or not. The resulting tree is called a *tree of ortholog groups* (or *TOGs*). MultiMSOAR 2.0 then employs one of the two algorithms devised in this paper (called the *NodeCentric* and *TreeCentric* algorithms) to label the internal nodes of each TOG based on the parsimony principle and some biological constraints. Ortholog groups can then be trivially identified from each fully labeled TOG. The details of each of the main steps in Figure 2 are explained in the METHODS section. Note that each ortholog group found by MultiMSOAR 2.0 is contained in some TOG but a TOG may contain several ortholog groups. An example is shown in Figure 1(g), where the TOG contains two ortholog groups and the second ortholog group contains a gene loss.

2. METHODS

2.1. Homology Search and Gene Family Construction

Since we have multiple genomes, we define a gene family to consist of all homologous genes on all the genomes under study. As in Ref. [5, 19], only protein coding genes will be considered. To cluster all the genes into gene families, we combine all protein sequences from all genomes together, and perform an all-vs-all BLASTp homology search[21]. Then we use the popular clustering program MCL[22] to construct gene families. Similar methods have been used in many other papers[7, 10, 23].

2.2. Pairwise Genome Comparison

Since we try to identify ortholog groups among multiple genomes based on pairwise comparison, the prediction accuracy of ortholog pairs between two genomes is critical for the performance of our multiple genome system. MSOAR 2.0 has shown to be the most accurate prediction tool for assigning one-to-one ortholog pairs between two closely related genomes[5]. So, it is preferable to use the output of MSOAR 2.0 as the input to our current system. For a comparison among S genomes, we apply MSOAR 2.0 to all pairs of the S genomes, and use the $S*(S-1)/2$ pairwise comparison results to define a multipartite for each gene family to be partitioned in MultiMSOAR 2.0.

2.3. Partition of Each Gene Family into TOGs

In our definition of ortholog groups, each group may include at most one gene from each genome. However, a gene family may include many homologous

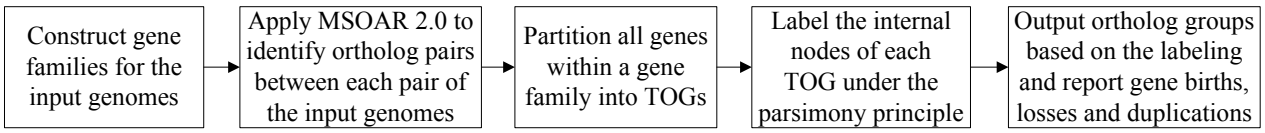


Fig. 2. An outline of MultiMSOAR 2.0.

genes from each genome (*i.e.*, paralogs), making it necessary to split the genes in a family into TOGs, such that each TOG contains at most one gene from every genome. This is done by employing a heuristic maximum weight S-dimensional matching algorithm as follows. Similar methods have been used in Ref. 20, 24.

Suppose we have S genomes, $G_1, G_2, ..., G_S$, where $S \geq 3$. For a given gene family, the number of genes from each genome are denoted as $n_1, n_2, ..., n_S$. We can construct an S-partite (or S-stage) graph G with n_i $(1 \leq i \leq S)$ vertices in the part corresponding to genome G_i (called stage i). We add edges to G by using the pairwise orthology information produced by MSOAR 2.0. Specifically, we add an edge between two vertices in G if and only if the corresponding two genes are from two different genomes and they are assigned as an ortholog pair by MSOAR 2.0. We assign a weight to such an edge, which is the BLASTp similarity score between the ortholog pair.

Since we would like to obtain a perfect S-dimensional matching with the maximum weight among the S stages, we need to add some dummy vertices to some of the stages in G to make them all have the same number of vertices. Let $N = \max_{1 \leq i \leq S} n_i$ be the maximum number of paralogs on any genome in the gene family. Then we add $N - n_i$ $(1 \leq i \leq S)$ dummy vertices to the i-th stage. The maximum (S-dimensional) matching problem for S-partite graphs (where $S \geq 3$) is known to be NP-hard[25], and N could be large for a real gene family when a large number of genomes are considered. So, we will use a heuristic optimization approach to find a good matching. Since the maximum weight matching for a bipartite graph can be computed by the Hungarian algorithm in cubic time[26], we first find a maximum weight bipartite matching for two stages in G, combine them into one stage, and apply the Hungarian algorithm iteratively on the remaining stages in G until only one stage is left. This results in a matching for the original S-partite graph G. This approach is very similar to the method used in MultiMSOAR[20], except that we use a post-order traversal on the species tree to decide the order that stages are combined. This way, a stage is always combined with another stage that is close to it on the species tree. Another difference is that we use the bit score as the weight of an edge in G. If there is no edge between two vertices in different stages, we deem that there is an edge with weight 0 between them.

An example of the gene family partition is shown in Figure 1, where the figures in (c), (d), (e) represent 3 TOGs for the α gene family while Figure 1(f) represents a single TOG for the β gene family.

2.4. Labeling of TOGs

In order to identify ortholog groups within a TOG, we need to label the internal nodes (which correspond to ancestral genomes) using binary representations as well. Here, 1 means that the a gene is present in the corresponding ancestral genome while 0 means absence. Two constraints will be assumed:

(1) *Intratree constraint*: If node u is labeled with a 0 and u has an ancestral node that is labeled with a 1, then every descendant node of u must be labeled with a 0.
(2) *Intertree constraint*: Suppose that u and v are two nodes such that each of them is labeled with a 1 in at least one TOG. Then every node on the path connecting u and v must be labeled with a 1 in at least one TOG.

The intertree constraint makes sure that no gene is born twice in evolution, which is a commonly accepted hypothesis in molecular evolution since double gene birth events are extremely rare. The intratree constraint follows from the definition of orthology (that orthologs evolved through speciation only).

Among all the labelings of the TOGs satisfying the above two constraints, we would like to find one that minimizes the number of gene births, duplications and losses in the evolution of the family. Since each edge of a TOG whose nodes are labeled with 01 or 10 represents a gene birth/duplication or a gene loss, we need to find a parsimonious way to label the internal nodes so that the number of 01 or 10 edges is minimized. For simplicity, let us call a 01 or 10 change on an edge a *flip*.

We can now formulate the TOG labeling problem as a combinatorial optimization problem as follows:

TOG Labeling: *Given N TOGs, find a binary labeling of all the internal nodes of the TOGs so that both intratree and intertree constraints are satisfied and the total number of flips is minimized.*

The problem can be solved by a trivial exhaustive search algorithm that considers all possible labelings of the TOGs. However, since a binary tree with S leaves has $S-1$ internal nodes, this algorithm runs in time $O(2^{N\cdot(S-1)})$, which is impractical even if $N = S = 10$. We need to find more efficient solutions to this problem.

Before we proceed with our algorithms, we first prove the following two lemmas, which will help accelerate the speed of our labeling algorithm.

Lemma 1. *If two child nodes are labeled as* 1*, then in any optimal labeling, their parent node must be labeled as* 1*.*

Proof. Suppose that in an optimal labeling L, an internal node P is labeled as 0 in some TOG but both of its children are labeled as 1. If we change the label of P to 1, the two constraints will not be violated, and there will be two fewer flips on the two edges from P to its two children. Even if this change might incur a new flip on the edge from P to its parent node, the total number of flips will still be reduced. This is a contradiction to the assumption that L is an optimal labeling, which completes our proof. □

Lemma 2. *If two child nodes are labeled as* 0*, then there is an optimal labeling, where their parent node is labeled as* 0*.*

Proof. Suppose that an internal node P of some TOG T is labeled as 1 while both of its children are labeled as 0 in some optimal labeling. If we change the label of P to 0, it is easy to see that the intratree constraint will not be violated. However, the intertree constraint might be violated if the node P is also labeled as 0 in all other TOGs. Then, according to Lemma 1, the two child nodes of P cannot be labeled as 1 at the same time in each of the other TOGs. If each of the two child nodes of P is labeled as 0 in all other TOGs, then we are safe to change the label of P from 1 to 0 in the TOG T since the change will not violate the intertree constraint. Otherwise, there is at least one TOG T', in which the two child nodes of P are labeled as 0 and 1, respectively. In this case, we can change the label of P in T' to 1. From the proof of Lemma 1, we know that changing the label of P in T will decrease the number of flips by at least 1, while changing the label of P in T' may increase the number of flips by at most 1. If we change the labels of node P is TOGs T and T' simultaneously, the total number of flips will not increase and thus the labeling is still optimal. Moreover, such a simultaneous change will keep the intertree constraint satisfied. This completes the proof of Lemma 2. □

The TOG labeling problem is trivial to compute without the intratree and intertree constraints. If we only consider the intratree constraint, the problem can still be solved by using dynamic programming in polynomial time. However, the intertree constraint makes the problem much harder. Here, we propose two different algorithms to solve the TOG labeling problem: the *NodeCentric* algorithm and the *TreeCentric* algorithm. The algorithms are sketched below.

The basic idea behind the NodeCentric algorithm is to label all N TOGs simultaneously by dynamic programming. In other words, it labels each internal node of the species tree with a binary vector of N bits. In order to keep track of the validity of the two constraints, we will use label $0'$ (when considering some TOG) to indicate that (i) the current node is labeled as 0 in the TOG and (ii) some descendant of the current node is labeled as 1 in the TOG. Thus, the label 0 now means that all descendant nodes are also labeled as 0. The algorithm proceeds in postorder. For each internal node u in the species tree, it enumerates all possible label vectors at u and for each vector, it computes the minimum number of flips in the subtree under node u by considering all feasible label vectors of its two children without violating the two constraints. By Lemmas 1 and 2, we can quickly fix the label of u in a TOG if the labels of its two children in the same TOG are both fixed as 0 or both fixed as 1.

Since the left and right children can be considered separately, it seems that the above algorithm would run in $O(S\cdot(3^N\cdot 3^N)) = O(S\cdot 9^N)$ time, which could be impractical if N is large. However, with a

careful analysis, we find that at most 3 (instead of 9) combinations of the parent-child labels are possible in a TOG. If the parent label is fixed as 0, then the child label must be fixed as 0 as well. Otherwise, the parent label could be $0'$ or 1. If it is $0'$, then the child label could be either fixed as 0 or one of $0'$ and 1. If the parent label is 1, then the child label must be fixed either as 0 or as 1 due to the intratree constraint. So, in any case, at most 3 combinations of the parent-child labels should be considered in a TOG and hence, a total number of 3^N values need to be computed. The intertree constraint may reduce the number of legal combinations even further. This implies an efficient implementation of the NodeCentric algorithm with time complexity $O(S \cdot 3^N)$.

While the NodeCentric algorithm goes through each node sequentially, the TreeCentric algorithm goes through each TOG sequentially. For a subset of fully labeled TOGs on the same species tree, the *union TOG* is a fully labeled TOG obtained by taking the Boolean *or* operation on the labels of each given TOG at the same node of the species tree. Let us order the TOGs arbitrarily as $T_1, T_2, \ldots, T_N$. For each T_i, the TreeCentric algorithm enumerates all possible union TOGs covering $T_1, T_2, \ldots, T_{i-1}$. For each such union TOG T^{i-1}, it enumerates all feasible binary labelings of the TOG T_i by taking into account the intratree constraint. It also records the number of flips in TOG T_i for each such feasible labeling of T_i. This second enumeration process can be done efficiently by dynamic programming. Then it computes and records the minimum number of flips in the TOGs $T_1, T_2, \ldots, T_i$ for each union TOG T^i covering $T_1, T_2, \ldots, T_i$, by taking advantage of the previously recorded minimum number of flips in $T_1, T_2, \ldots, T_{i-1}$ for each union TOG T^{i-1}. Finally, the minimum number of flips in all TOGs $T_1, T_2, \ldots, T_N$ is obtained by considering all possible union TOGs covering $T_1, T_2, \ldots, T_N$ and taking into account the intertree constraint. Since the number of different union TOGs is 2^{S-1}, the above algorithm runs in $O(N \cdot 4^{S-1})$ time.

Both the NodeCentric and TreeCentric algorithms are exponential time algorithms. However, in practice, the number of genomes in comparison is expected to be small (usually $S \leq 15$). So we can use the TreeCentric algorithm to find an optimal TOG labeling efficiently. When the value of N is smaller, it is faster to apply the NodeCentric algorithm. Note that, the two algorithms may find different labelings for the same input, both of which are optimal.

2.5. Ortholog Group Identification

After labeling all TOGs, it is straightforward to identify ortholog groups. Starting from the root of each TOG, we can find the highest ancestral nodes labeled as 1. All genes at the descendent leaves of such an ancestral node form an ortholog group. An example is shown in Figure 1(g). In addition, with the labeling of each TOG, we can easily identify evolutionary events including gene births and losses as well as duplications. For each edge in the TOG, if the parent-child labeling is 1-0, then there is a gene loss. If the labeling is 0-1, and the parent node is labeled as 0 in all other TOGs, then it represents a gene birth. Otherwise, it represents a gene duplication.

3. EXPERIMENTS AND RESULTS

In order to test the performance of our system MultiMSOAR 2.0, we first apply it to simulated data, and compare it with the popular ortholog assignment tool MultiParanoid[27] for multiple genomes. For real data experiments, besides comparison with MultiParanoid, we also compare our results with Roundup[28], which is a well known multi-genome repository of orthology information.

3.1. Simulation Results

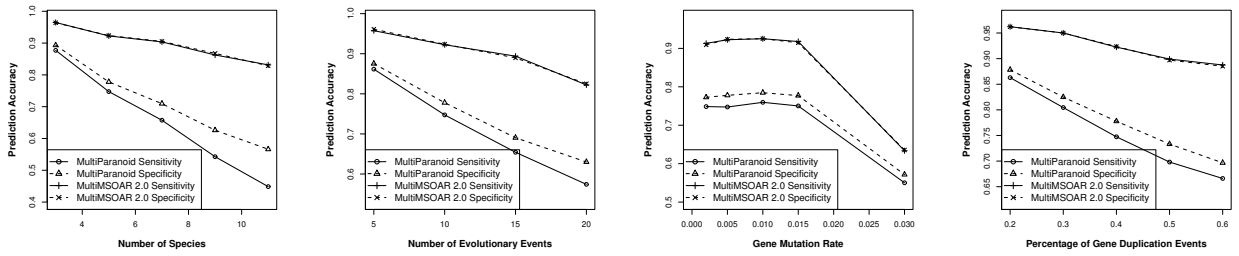
Our simulation test is an extension of the one in Ref. 5 for testing the performance of MSOAR 2.0. However, we now need to simulate more genome evolutionary events, including gene mutations, gene births, gene duplications, gene losses, genome rearrangements (including reversals, translocations, fusions and fissions) and speciations. To make things easier, we only simulate the evolution of S ($S \leq 15$) single-chromosomal genomes as done in Ref. 5. In order to generate S contemporary genomes, we first generate a random species tree T with S leaf nodes. Each internal node in T represents an ancestral genome while the leaf nodes represent the current genomes. Each edge in T represents a speciation

event. We then randomly generate a genome with 100 genes consisting of 3,000 nucleotides each at the root of T. For each speciation event, we simulate E evolutionary events, which include α gene duplications, β gene births, γ gene losses, and $(1-\alpha-\beta-\gamma)$ genome rearrangements. To generate the gene duplications, we randomly choose a gene, copy it and insert it into the genome next to the original copy or at a random position, depending on whether the duplication is tandem or random (here we assume 50% of all duplications are tandem, as done in Ref. 5). To simulate the birth of a new gene, we create a new gene and randomly insert it into the genome. To simulate the loss of a gene, we randomly choose a gene and delete it from the genome. For genome rearrangements, since there is only one chromosome, only reversals are considered. Reversals are simulated by randomly choosing two positions on the genome and reverse all the genes between them. To simulate gene (point) mutations, we set a gene mutation rate μ to allow all the genes on all the existing genomes to have μ mutations between every two evolutionary events. In other words, a molecular clock is assumed. In summary, our simulation data is controlled by a 6-parameter set: $(S, E, \mu, \alpha, \beta, \gamma)$, where S is the number of species, E the total number of evolutionary events after each speciation, μ the gene mutation rate, and α, β, γ the percentages of gene duplications, births and losses among the E events, respectively.

To study the effects of different parameters on the performance of MultiMSOAR 2.0, we set the default values for each parameter as $S = 5, E = 10, \mu = 0.5\%, \alpha = 40\%, \beta = 10\%, \gamma = 10\%$, and we will vary one parameter at a time. To measure the prediction accuracy, we use two popular measurements: *sensitivity* and *specificity*. Here, sensitivity is defined as the number of the true ortholog groups (*i.e.*, true positives) identified by a program divided by the total number of known ortholog groups, and specificity is defined as the number of true ortholog groups identified divided by the number of ortholog groups output. We compare the ortholog groups found by MultiMSOAR 2.0 and MultiParanoid. In order for an identified ortholog group to be a true positive (*i.e., TP*), we require that all genes in the identified ortholog group match exactly with all the genes in a known ortholog group. For each parameter set, we generate 10 simulated data sets and run MultiMSOAR 2.0 and MultiParanoid on these data respectively. Finally we calculate the average prediction accuracies of the two programs on each parameter set. The prediction accuracies of the two programs are shown in Figures 3.

Figures 3(a), (b), (d) show that with the increase of the number of species, the number of evolutionary events, and the number of gene duplications, the prediction accuracies of both programs decrease since it becomes harder for them to correctly identify ortholog groups. However, we notice that the decrease in accuracy for MultiMSOAR 2.0 is mild while the decrease is sharp for MultiParanoid, especially in Figure 3(d). This could be because when more genes are duplicated, it becomes increasingly difficult for MultiParanoid to decide if a duplication happened in an ancient genome or in a more recent genome. Thus, it might confuse some ancient duplications with recent duplications and miss calling some true ortholog groups. On the other hand, MultiMSOAR 2.0 infers the time of each duplication explicitly when labeling TOGs, and is thus more resilient to the increase of gene duplication events. However, since the labeling algorithm used in MultiMSOAR 2.0 is based on the parsimony principle and the optimal labeling might not be unique, the actual labeling given by MultiMSOAR 2.0 may not necessarily reflect the true evolutionary history. As a result, when the number of gene duplications increases, the prediction accuracy of MultiMSOAR 2.0 also decreases, but much more slowly than in the case of MultiParanoid.

Figure 3(c) is very interesting. With the increase of gene mutation rate μ from 0.2% to 1%, the sensitivities of both programs and the specificity of MultiMSOAR 2.0 increase a little bit. This is because when μ increases, it becomes slightly easier for both programs to differentiate duplicated genes from their original copies based on sequence similarity. However, when μ goes from 1% to 3%, the prediction accuracies of both programs sharply decrease. This is because the sequence similarity between homologous genes originated from a common ancestral gene becomes weaker with the increase of μ. As a result, it becomes harder for MultiParanoid to identify ortholog groups solely based on sequence similarity,



(a) (b) (c) (d)

Fig. 3. Comparison of MultiMSOAR 2.0 and MultiParanoid on simulated data. (a) Simulation results on the parameter set $(*, 10, 0.5\%, 40\%)$ where the parameter S is varied. (b) Simulation results on the parameter set $(5, *, 0.5\%, 40\%)$ where the parameter E is varied. (c) Simulation results on the parameter set $(5, 10, *, 40\%)$ where the parameter μ is varied. (d) Simulation results on the parameter set $(5, 10, 0.5\%, *)$ where the parameter α is varied.

and for the MCL algorithm used in MultiMSOAR 2.0 to correctly cluster homologous genes into a gene family. Without correct gene families, we cannot expect MultiMSOAR 2.0 to find the ortholog groups correctly.

Generally speaking, from the four figures above, we can see that the prediction accuracy of MultiMSOAR 2.0 is significantly higher than that of MultiParanoid. With more species, more evolutionary events and more gene duplications, the advantage of MultiMSOAR 2.0 over MultiParanoid becomes more apparent. Besides, in the simulation, MultiMSOAR 2.0 is always able to achieve at least 80% prediction accuracy (in terms of sensitivity and specificity) as long as the gene mutation rate is not too high. This is pretty remarkable considering the large number of species and evolutionary events involved. Moreover, MultiMSOAR 2.0 can provide much more information about gene births, losses and duplications in addition to identifying ortholog groups. In the simulation tests, MultiMSOAR 2.0 is able to predict gene birth, loss and duplication events with an accuracy more than 60% in general. Due to the page limit, the prediction accuracies concerning these events by MultiMSOAR 2.0 on simulated data are summarized in Tables 3-6 given in the Appendix although a detailed discussion is omitted.

3.2. Real Data Experiments

Since MultiMSOAR 2.0 is a tool to identify ortholog groups for multiple genomes that are closely related on a genome scale, to test its performance on real data, we choose to use the mammalian genomes that have been completely sequenced. We downloaded seven mammalian genomes from the Ensembl genome browser (http://www.ensembl.org/): human (*Homo sapiens*), chimpanzee (*Pan troglodytes*), macaque (*Macaca mulatta*), mouse (*Mus musculus*), rat (*Rattus norvegicus*), cow (*Bos taurus*) and opossum (*Monodelphis domestica*) (version 57, March 2010). The species tree for the seven mammalian genomes is downloaded from Ensembl as well.

For the purpose of comparison, we choose to compare the results of MultiMSOAR 2.0 with those of the popular tool MultiParanoid and ortholog database Roundup. For MultiParanoid, we deem all the genes in the same cluster output by the program as an ortholog group assigned by MultiParanoid. We run MultiMSOAR 2.0 and MultiParanoid on the real data sets respectively and compare their results. Roundup is a recently developed multi-genome repository of orthologs for over 250 genomes[28]. We download the ortholog groups for the concerned genomes from its website (http://roundup.hms.harvard.edu/). Since Roundup uses genomes from different sources, we need to map the genes used in Roundup to the corresponding genes used in Ensembl.

Some other tools and databases are also available for ortholog assignment among multiple genomes, such as the OrthoFocus program, and the Ensembl ortholog database. However, OrthoFocus is a program to identify orthologs in family-focused studies and it is inapplicable to genome-scale comparisons[29].

We tried to compare our results with the well known Ensembl ortholog database[3]. However, we found the comparison inappropriate. First of all, Ensembl requires the user to specify a "center" genome based on which a multi-genome orthology information will be retrieved. As a result, the orthology information provided by Ensembl might miss many ortholog groups that do not involve the center genome. Second, it only supports queries of orthology information for up to 5 species, which is quite limited compared to the large number of completely sequenced genomes. Third, it generally presents orthology as a many-to-many relationship and when more genomes are being compared, it uses a single-linkage method to combine ortholog groups. Thus, an ortholog group defined by Ensembl may include many paralogous genes from each genome. Since we define orthology as a one-to-one relationship, it would be difficult for us to fairly compare the results of MultiMSOAR 2.0 with those in the Ensembl ortholog database.

3.2.1. *Results on Human, Mouse and Rat*

Since human, mouse and rat are the best annotated genomes, we can use gene symbols to validate the ortholog groups assigned among the three genomes by different programs. The same validation method has been used in many other papers[5, 19, 20]. By using gene symbols, we can define true ortholog groups (TPs), false ortholog groups (FPs), and unknown ortholog groups as follows. If an ortholog group contains genes that have different gene symbols, then this group is counted as an FP. If at most one of the genes in the group have gene symbols, then this group is counted as an unknown. Otherwise, we treat the group as a TP. An ortholog group is defined as *assignable* if its genes appear in at least two genomes and have exactly the same gene symbol. We use the same measurements *sensitivity* and *specificity* as defined in the simulation to measure the prediction accuracies of the three programs. The performance of the programs is shown in Table 1.

The low sensitivity of Roundup in Table 1 may be caused by the mapping of gene IDs from Roundup to Ensembl since quite a few of the genes in Roundup were mapped to the unknowns in Ensembl. Nevertheless, we can see that MultiMSOAR 2.0 achieves the best sensitivity and specificity among all three programs. This is mainly because MultiParanoid only considers sequence similarity when assigning ortholog groups. Though Roundup is based on the reciprocal smallest distance algorithm, which is different from the reciprocal BLAST hits used in MultiParanoid, it fails to consider other information as well. In contrast, MultiMSOAR 2.0 combines gene order with sequence similarity and thus is able to make more accurate predictions.

3.2.2. *Results on All Seven Mammalian Genomes*

When comparing the seven mammalian genomes including human, chimpanzee, macaque, mouse, rat, cow, and opossum, we cannot validate the ortholog groups predicted by the three programs using gene symbols since not all of the genomes have been annotated with gene symbols. So, we only consider the common and different ortholog groups assigned by MultiMSOAR 2.0, MultiParanoid and Roundup. The comparison results are shown in Table 2.

Table 2 shows the numbers of ortholog groups involving 2 to 7 genomes that were identified by the three programs. From Table 2, we can see that the numbers of ortholog groups found by MultiMSOAR 2.0 and MultiParanoid are close to each other for each number of genomes involved while they differ a lot from those of Roundup. Again, this might be caused by the mapping from the data used in Roundup repository to the data used in Ensembl. Nevertheless, MultiMSOAR 2.0 identified the most number of ortholog groups involving all seven genomes, and it shares 9,075 common ortholog groups with MultiParanoid, which provides an indirect support of the ortholog groups found by MultiMSOAR 2.0. Moreover, we find that the closer the genomes contained in an ortholog group are in the species tree, the more likely it is shared between MultiMSOAR 2.0 and MultiParanoid (actual data not shown). The large number of ortholog groups involving all seven genomes found by both MultiMSOAR 2.0 and MultiParanoid is a manifest of the evolutionary closeness of the seven mammalian species. The number of ortholog groups involving 4 genomes found by both programs is pretty small here, since there is no subtree in the species tree consisting of

Table 1. Performance of the three programs on human, mouse and rat.

Program	Assignable TPs	TPs	FPs	Unknowns	Total	Sensitivity	Specificity
MultiMSOAR 2.0	15,598	14,051	2,399	2,919	19,369	90.08%	85.42%
MultiParanoid	15,598	13,697	2,609	2,328	18,634	87.81%	84.00%
Roundup	14,616	10,094	2,424	6,790	19,308	69.06%	80.66%

Table 2. Ortholog groups shared by the three programs on the seven mammalian genomes.

Programs	7 genomes	6 genomes	5 genomes	4 genomes	3 genomes	2 genomes
MultiMSOAR 2.0	12,034	3,772	1,337	584	875	3,195
MultiParanoid	11,397	3,311	1,127	609	800	2,728
Roundup	4,294	5,574	3,965	2,098	1,475	2,720
MultiMSOAR 2.0 and MultiParanoid	9,075	2,237	633	239	348	1,483
MultiMSOAR 2.0 and Roundup	3,122	610	120	49	96	353
MultiParanoid and Roundup	2,676	532	127	80	121	429
All three programs	2,614	464	103	35	68	254

exactly four species. Hence, an ortholog group of size four would have to involve gene losses. Since there is only one subtree consisting of three species (*i.e.*, human, chimpanzee, and macaque), most of the 875 ortholog groups of size 3 found by MultiMSOAR 2.0 (679, or about 77.6%) consist of genes from the three species. Similarly, 1,772/3,195 (55.46%) and 1,083/3,195 (32.49%) of the ortholog groups of size two consist of genes from mouse-rat and human-chimpanzee respectively, both of which are the closest pairs in the species tree.

4. CONCLUSION AND DISCUSSION

In this paper, we have extended the pairwise ortholog assignment system MSOAR 2.0 to a multi-genome ortholog assignment system MultiMSOAR 2.0. By comparing with the well known multi-genome ortholog assignment tool MultiParanoid on simulated data, we demonstrated that MultiMSOAR 2.0 achieves a significantly higher prediction accuracy. Our real data experiments on seven closely related mammalian genomes also show the superior performance of MultiMSOAR 2.0 over MultiParanoid and the multi-genome ortholog repository Roundup. Moreover, not only can MultiMSOAR 2.0 identify ortholog groups accurately, it can also provide accurate information about gene births, losses and duplications, which may shed additional insight on genome evolution.

ACKNOWLEDGEMENTS

We would like to thank Liqing Zhang for many constructive discussions. This work was supported in part by National Science Foundation grant IIS-0711129.

References

1. W. M. Fitch. Distinguishing homologous from analogous proteins. *Syst Zool*, 19(2):99–113, June 1970.
2. Z. Jiang, J. Michal, J. Melville, and H. Baltzer. Multi-alignment of orthologous genome regions in five species provides new insights into the evolutionary make-up of mammalian genomes. *Chromosome Research*, 13(7):707–715, 2005.
3. A. J. Vilella, J. Severin, et al. EnsemblCompara GeneTrees: Complete, duplication-aware phylogenetic trees in vertebrates. *Genome Research*, 19(2):327–335, 2009.
4. M. Remm, C. E. Storm, and E. L. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *Journal of Molecular Biology*, 314(5):1041–1052, 2001.
5. G. Shi, L. Zhang, and T. Jiang. MSOAR 2.0: Incorporating tandem duplications into ortholog assignment based on genome rearrangement. *BMC Bioinformatics*, 11(1):10, January 2010.
6. R. L. Tatusov, D. A. Natale, et al. The COG database: new developments in phylogenetic classification of proteins from complete genomes. *Nucleic Acids Research*, 29(1):22–28, 2001.
7. L. Li, C. J. Stoeckert, and D. S. Roos. OrthoMCL: Identification of ortholog groups for eukaryotic genomes. *Genome Research*, 13(9):2178–2189, 2003.
8. A. C. Berglund, E. Sjölund, et al. InParanoid 6: eukaryotic ortholog clusters with inparalogs. *Nucleic Acids Research*, 36(Database issue), January 2008.
9. D. L. Wheeler, T. Barrett, et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 34(suppl-1):D173–180, 2006.
10. H. Li, A. Coghlan, J. Ruan, et al. TreeFam: a curated database of phylogenetic trees of animal gene

families. *Nucleic Acids Research*, 34(suppl-1):D572–580, 2006.
11. L. Goodstadt and C. P. Ponting. Phylogenetic reconstruction of orthology, paralogy, and conserved synteny for dog and human. *PLoS Comput Biol*, 2(9):e133, 09 2006.
12. M. V. Han and M. W. Hahn. Identifying parent-daughter relationships among duplicated genes. *Pacific Symposium on Biocomputing*, pages 114–125, 2009.
13. F. Boyer, A. Morgat, L. Labarre, et al. Syntons, metabolons and interactons: an exact graph-theoretical approach for exploring neighbourhood between genomic and functional data. *Bioinformatics*, 21(23):4209–4215, 2005.
14. Y. P. Deniélou, F. Boyer, M. F. Sagot, and A. Viari. Recovering isofunctional genes: a synteny-based approach. *Actes des Journée Ouvertes de Biologie, Informatique et Mathématiques*, 2008.
15. S. Hannenhalli and P. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, Washington, DC, USA, 1995. IEEE Computer Society.
16. W. J. Kent, R. Baertsch, et al. Evolution's cauldron: Duplication, deletion, and rearrangement in the mouse and human genomes. *Proceedings of the National Academy of Sciences of the United States of America*, 100(20):11484–11489, 2003.
17. P. Pevzner and G. Tesler. Genome rearrangements in mammalian evolution: Lessons from human and mouse genomes. *Genome Research*, 13(1):37–45, 2003.
18. M. Semon and K. H. Wolfe. Rearrangement rate following the whole-genome duplication in teleosts. *Molecular Biology and Evolution*, 24(3):860–867, 2007.
19. Z. Fu, X. Chen, V. Vacic, et al. MSOAR: A high-throughput ortholog assignment system based on genome rearrangement. *Journal of Computational Biology*, 14(9):1160–1175, 2007.
20. Z. Fu and T. Jiang. Clustering of main orthologs for multiple genomes. *Journal of Bioinformatics and Computational Biology*, 6(3):573–584, 2008.
21. S. F. Altschul, W. Gish, W. Miller, et al. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, October 1990.
22. A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, 2002.
23. V. Shoja and L. Zhang. A roadmap of tandemly arrayed genes in the genomes of human, mouse, and rat. *Molecular Biology and Evolution*, 23(11):2134–2141, 2006.
24. A. Vashist, C. A. Kulikowski, and I. Muchnik. Ortholog clustering on a multipartite graph. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 4(1):17–27, 2007.
25. V. Kann. Maximum bounded 3-dimensional matching is max snp-complete. *Inf. Process. Lett.*, 37(1):27–35, 1991.
26. H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics*, 52(1):7–21, 2005.
27. A. Alexeyenko, I. Tamas, G. Liu, and E. L. L. Sonnhammer. Automatic clustering of orthologs and inparalogs shared by multiple proteomes. *Bioinformatics*, 22(14):e9–15, July 2006.
28. T. F. DeLuca, I. Wu, J. Pu, et al. Roundup: a multi-genome repository of orthologs and evolutionary distances. *Bioinformatics*, 22(16):2044–2046, August 2006.
29. A. E. Ivliev and M. G. Sergeeva. OrthoFocus: program for identification of orthologs in multiple genomes in family-focused studies. *Journal of Bioinformatics and Computational Biology*, 6(4):811–824, August 2008.

Appendix

Prediction Accuracy of MultiMSOAR 2.0 on Gene Births, Losses and Duplications on Simulated Data.

Table 3. Prediction accuracy when the parameter S (the number of species) is varied.

S	3	5	7	9	11
GeneBirth Sensitivity	97.50%	100.0%	95.83%	97.50%	87.50%
GeneBirth Specificity	72.17%	81.78%	88.17%	89.25%	97.87%
GeneDuplication Sensitivity	79.38%	77.19%	75.83%	74.06%	65.38%
GeneDuplication Specificity	84.95%	79.16%	81.35%	77.22%	79.70%
GeneLoss Sensitivity	42.50%	63.75%	63.03%	71.00%	79.14%
GeneLoss Specificity	66.67%	73.89%	78.29%	79.41%	45.00%

Table 4. Prediction accuracy when the parameter E (the number of evolutionary events) is varied.

E	5	10	15	20
GeneBirth Sensitivity	100.0%	100.0%	98.75%	97.50%
GeneBirth Specificity	83.73%	81.78%	80.83%	84.26%
GeneDuplication Sensitivity	79.38%	77.19%	73.96%	65.00%
GeneDuplication Specificity	81.82%	79.16%	79.38%	73.12%
GeneLoss Sensitivity	65.50%	63.75%	56.43%	52.58%
GeneLoss Specificity	78.00%	73.89%	62.44%	71.50%

Table 5. Prediction accuracy when the parameter μ (gene mutation rate) is varied.

μ	0.2%	0.5%	1%	1.5%	3%
GeneBirth Sensitivity	96.25%	100.0%	98.75%	98.75%	97.92%
GeneBirth Specificity	78.67%	81.78%	80.67%	74.05%	51.05%
GeneDuplication Sensitivity	71.88%	77.19%	78.44%	77.81%	77.60%
GeneDuplication Specificity	77.64%	79.16%	81.54%	80.27%	90.78%
GeneLoss Sensitivity	53.75%	63.75%	63.75%	62.14%	54.76%
GeneLoss Specificity	64.31%	73.89%	83.29%	73.28%	85.12%

Table 6. Prediction accuracy when the parameter α (the ratio of duplication events) is varied.

α	20%	30%	40%	50%	60%
GeneBirth Sensitivity	98.75%	98.75%	100.0%	97.50%	96.25%
GeneBirth Specificity	82.16%	82.53%	81.78%	82.31%	80.75%
GeneDuplication Sensitivity	78.12%	80.00%	77.19%	72.00%	73.54%
GeneDuplication Specificity	84.09%	83.82%	79.16%	79.38%	81.26%
GeneLoss Sensitivity	66.61%	65.00%	63.75%	55.00%	52.50%
GeneLoss Specificity	87.98%	85.86%	73.89%	69.27%	52.64%