Matching Algorithms for Assigning Orthologs after Genome Duplication Events

Guillaume Fertin[a,1], Falk Hüffner[2], Christian Komusiewicz[b,3], Manuel Sorge[c,d,4]

[a] *LS2N UMR CNRS 6004, University of Nantes, Nantes, France.*
[b] *Fachbereich für Mathematik und Informatik, Philipps-Universität Marburg, Marburg, Germany.*
[c] *Ben-Gurion University of the Negev, Beer Sheva, Israel.*
[d] *Technische Universität Berlin, Berlin, Germany.*

Abstract

In this paper, we introduce and analyze two graph-based models for assigning orthologs in the presence of whole-genome duplications, using similarity information between pairs of genes. The common feature of our two models is that genes of the first genome may be assigned two orthologs from the second genome, which has undergone a whole-genome duplication. Additionally, our models incorporate the new notion of *duplication bonus*, a parameter that reflects how assigning two orthologs to a given gene should be rewarded or penalized. Our work is mainly focused on developing exact and reasonably time-consuming algorithms for these two models: we show that the first one is polynomial-time solvable, while the second is NP-hard. For the latter, we thus design two fixed-parameter algorithms, i.e. exact algorithms whose running times are exponential only with respect to a small and well-chosen input parameter. Finally, for both models, we evaluate our algorithms on pairs of plant genomes. Our experiments show that the NP-hard model yields a better cluster quality at the cost of lower coverage, due to the fact that our instances cannot be completely solved by our algorithms. However, our results are altogether encouraging and show that our methods yield biologically significant predictions of orthologs when the duplication bonus value is properly chosen.

Email addresses: `guillaume.fertin@univ-nantes.fr` (Guillaume Fertin), `komusiewicz@informatik.uni-marburg.de` (Christian Komusiewicz), `sorge@post.bgu.ac.il` (Manuel Sorge)

[1]GF was supported by PHC Procope (project 17746PC).

[2]FH was supported by the DFG, project ALEPH (HU 2139/1).

[3]CK was supported by the DFG, project MAGZ (KO 3669/4-1), and by the DAAD Procope program (project 57317050).

[4]MS was supported by the DFG, project DAPA (NI 369/12), the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 631163.11, and the Israel Science Foundation (grant no. 551145/14).

1. Introduction

Identifying orthologous genes between two or more different species is a ubiquitous task in computational biology that plays an important role in phylogenetic tree inference, genome annotation and gene function prediction. Over the last few decades, a very large literature has been devoted to it – see among others e.g. [1–3] for recent surveys and benchmarking efforts on the subject.

A wide range of methods and accompanying software has been proposed to the community, which differ in many features, such as the combinatorial model, the optimization criteria, or the algorithmic strategies. In this paper, we focus our attention on graph-based models, and our goal in terms of resolution strategy is to provide methods that are both exact and reasonably time-consuming. For this, we chose, as a first step, to use an optimization criterion that only relies on sequence similarities between genes, similarly to e.g. [4, 5], although other methods also take further information into account such as relative positions of the genes [6–8].

Our work is initially based on work by Zheng et al. [5] who proposed the following two-step framework to identify groups of orthologous genes. First, compute a graph whose vertices correspond to the genes from the different species: an edge in the graph is present if the two corresponding genes are from different species and if it is plausible to consider these two vertices as orthologs. A simple way to build such a graph is to first compute sequence similarity scores between the genes of the two species and then add an edge between a pair of vertices if the corresponding sequence similarity exceeds a predefined threshold [9]. More sophisticated approaches take further information into account, obtained for example by synteny block construction [5] or by analyzing the protein interaction networks of the respective species [10]. The second step in the framework is to identify disjoint groups of orthologs such that, roughly speaking, there are many edges inside the identified groups and few edges between them. In the basic setting of Zheng et al. [5], each group of orthologs contains at most one gene from each species. However, this constraint is too strict in the presence of whole-genome duplications, which occur relatively frequently during plant evolution. Therefore, Zheng et al. [5] also considered a relaxed definition of orthology groups, where a predetermined subset of the species is allowed to have two genes in each orthology group.

The above-described model from Zheng et al. [5] is the starting point of this paper. More specifically, we focus our attention on the special case in which we aim to identify disjoint groups of orthologous genes from two species and where exactly one of the two species has been subject to a whole-genome duplication since the speciation event. We then extend this model in two directions: first, we introduce a new parameter d, the *duplication bonus*, that is applied each time where, in our solution, a gene is assigned two orthologous genes in the other species (a *1–2 orthology*). We call this new problem ORTHOLOGY ASSIGNMENT WITH DUPLICATIONS (OAD). Second, in addition to the duplication bonus, we now allow three different types of orthology clusters in a solution: 1-1 orthology, and two specific types of 1-2 orthologies that both require a minimum level of similarity between the genes of the two species. We call this second problem Orthology Clustering with Duplications (OCD). Our goal in the present work is twofold: first, to provide a deep algorithmic study of both problems that allows us to design exact and reasonably time-consuming (i.e., polynomial-time or fixed-parameter tractable) algorithms. Second, through several experiments on real data, to determine whether the proposed models are relevant, notably concerning the duplication bonus. Possible extensions of this model towards more elaborate ones are a natural follow-up of the present work.

Related Work. We first note that Orthology Assignment with Duplications, in the specific case where $d = 0$, is a particular case of the well-studied capacitated transportation problem (see for instance [11] for a formal definition). It can also be seen that it is equivalent to the Degree Constrained Subgraph (or DCS) problem, in which the input consists of an edge-weighted graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ and a set $I = \{[l_1, u_1], [l_2, u_2], \ldots, [l_n, u_n]\}$ of intervals such that for each $1 \leq i \leq n$ we have $l_i \leq u_i \leq d_G(v_i)$ (where $d_G(v_i)$ represents the degree of v_i in G). In the DCS problem we aim to find a maximum edge weighted subgraph G' of G such that for each $1 \leq i \leq n$ we have $l_i \leq d_{G'}(v_i) \leq u_i$. It is easy to see that, for any instance of our Orthology Assignment with Duplications problem with $d = 0$, we can transform it into an instance for DCS by taking the same graph G together with its edge-cost function, and by setting (i) $[l_i, u_i] = [0, 2]$ for each vertex representing a gene in the genome having undergone a whole-genome duplication event, and (ii) $[l_i, u_i] = [0, 1]$ for each vertex in the other genome. In that case, both problems have the same optimal solution. DCS has been shown to be polynomial-time solvable (see e.g. [12]). However, to our knowledge, the general case with an arbitrary duplication bonus $d \neq 0$ has not been studied before. The Orthology Clustering with Duplications problem is, to our knowledge, introduced for the first time in the present paper.

The complexity of related assignment problems for more than two genomes has been previously studied in the literature. The case of three genomes [5] is NP-hard even if duplications are not considered [13] – in this setting, one allows at most one gene of each genome to be in an orthologous group and one wants to maximize the number of edges that are in orthologous groups.

Zheng et al. [5] also proposed to assign the score for a cluster of orthologs based on its size. More precisely, each group of orthologs must induce a connected subgraph and the score is the number of edges in the transitive closure of this subgraph. For this scoring function, the computational problem to find a maximum-score orthology assignment is also NP-hard if the number of genomes is at least three [14, 15] and it is fixed-parameter tractable with respect to the score of the optimum solution [16]. The special case with two genomes, where the genes of exactly one of them may occur twice in each cluster of orthologs, is the special case of OAD with unit weights and duplication bonus $d = 1$. Adamaszek et al. [14] note that this special case can be solved using a matching algorithm without giving details.

Our Contribution. As mentioned above, this paper is primarily concerned with an algorithmic study of problems OAD and OCD. In Section 3, we show that OAD can be solved in $O(\sqrt{n} \cdot m)$ time for any duplication bonus. Since the basic idea is to reduce to maximum-weight matching problems, this means that improving the running time for OAD would imply an improved running time for maximum-weight matching in general graphs. In contrast, in Section 4 we show that OCD is NP-hard even if each edge weight is one. On the positive side, we give two fixed-parameter algorithms for OCD with running times $O(2^{k_B} \cdot (m + n))$ and $O(3^{k_W} \cdot \sqrt{n} \cdot m)$, respectively. Herein, n is the number of vertices of the graph representing our problem, m is its number of edges, and k_B, k_W are parameters that measure how far the instance is from certain types of polynomial-time solvable instances. Both parameters are smaller than the number k of edges that are discarded by an optimal solution. In Section 5, we experimentally run OAD and OCD for three plant species using the algorithms we have developed, and perform a preliminary comparison with InParanoid [4, 17–20], a well established graph-based orthology assignment software using similarity information between pairs of genes. Unsurprisingly, our results indicate that, in terms of execution time, OAD can be solved efficiently and that solving OCD is much more challenging as only parts of the instances can be solved exactly. For OCD, however, cluster quality is higher than for OAD. The experiments also show that using negative duplication bonus seems to give the best results in terms of biological quality of the output clusters of size three. Finally, the comparison to InParanoid indicates that our methods are much less conservative, identifying many more orthologs, but also including some of lower biological quality.

2. Notations and Definitions

As described in the previous section, the OAD and OCD problems start by representing genes as vertices of a graph and gene similarities as edges of this graph. More precisely, in OAD the graph at hand is a bipartite graph $G = (W, B, E)$. The vertices from the first part of the vertex set W correspond to the genes from one species, the vertices from the second part B correspond to the genes from the other species, and the task is to pair orthologs from different parts. In the classic graph-theoretical matching definition, each vertex is assigned at most one other vertex as a matching is a set of edges with disjoint endpoints. In order to model genome duplications, this is formally extended as follows.

Definition 1. *Let $G = (W, B, E)$ be a bipartite graph with partite vertex sets W and B. An* orthology assignment with duplications *in G is a set A of edges such that*

- *each vertex of W is an endpoint of at most* two *edges of A, and*
- *each vertex of B is an endpoint of at most* one *edge of A.*

For simplicity, we call the vertices in W (resp. B) white (resp. black) vertices, and we use the term "white gene" (resp. "black gene") to denote a gene represented by a vertex from W (resp. B). Similarly, the terms "white genome" and "black genome" will be used by abuse of notation. Since there can be other causes for gene duplications, for example tandem duplications, it is plausible, also in the presence of a whole genome duplication, that a white gene has more than two orthologs in the black genome and that black genes have more than one ortholog in the white genome. Allowing such larger clusters of orthologs has, however, the drawback that the orthology relations are not fully resolved [21]. Here, we wish to fully resolve orthology relations, whenever possible, with the only exception being paralogs caused by the genome duplication. Hence, the restriction of Definition 1.

In order to model that it is more likely that a gene of the white genome is matched to two black genes than to one, one may use a duplication bonus d. Moreover, one may assign weights to the edges to model that some gene pairs are more likely to be orthologs than others. With this in mind, we can formally define the Orthology Assignment with Duplications problem as follows.

Orthology Assignment with Duplications (OAD)
Instance: An undirected bipartite graph $G = (W, B, E)$, an edge-weight function $s\colon E \to \mathbb{Q}^+$, a duplication bonus $d \in \mathbb{Q}$.
Task: Find an orthology assignment with duplications, A, maximizing

$$\mathrm{val}(A) := \sum_{a \in A} s(a) + d \cdot |\{w \in W \mid w \text{ is an endpoint of two edges of } A\}|.$$

An example input instance with optimal solutions for different values of d is shown in Figure 1. We call val(A) the *value* of A. Observe that d can also be negative: in that case, matching a white gene to two black genes is penalized instead of encouraged, which might be useful, for example to avoid that the number of matched white genes is too low. Moreover, in the above model, any information about the similarity of genes in B is ignored.

A natural extension of this model is thus to incorporate knowledge about genes from the duplicated genome. More precisely, we can assume that the graph G contains, for each pair of black vertices, an edge if the corresponding genes are likely to be in-paralogs. Using a maximum-parsimony approach, the aim is then to identify clusters of orthologs that minimize the total weight of the edges between clusters as these contradict the orthology groups. Equivalently, we aim at finding clusters that maximize the weight of edges within the sought clusters. For the formal problem definition, first observe that, in this second model, the input graph is not bipartite anymore. Second, we need to define the set of allowed groups of orthologs. For this, from now on, let the term *cluster* denote a (connected) subgraph of G that contains at least one edge. Since the overall aim is still to find orthology relations between the white and black vertices, the clusters we allow in our model must contain only one white vertex, and must be connected to at least one and at most two black vertices. Among such possible clusters, we allow only three types, that we will call *ortholog clus-*

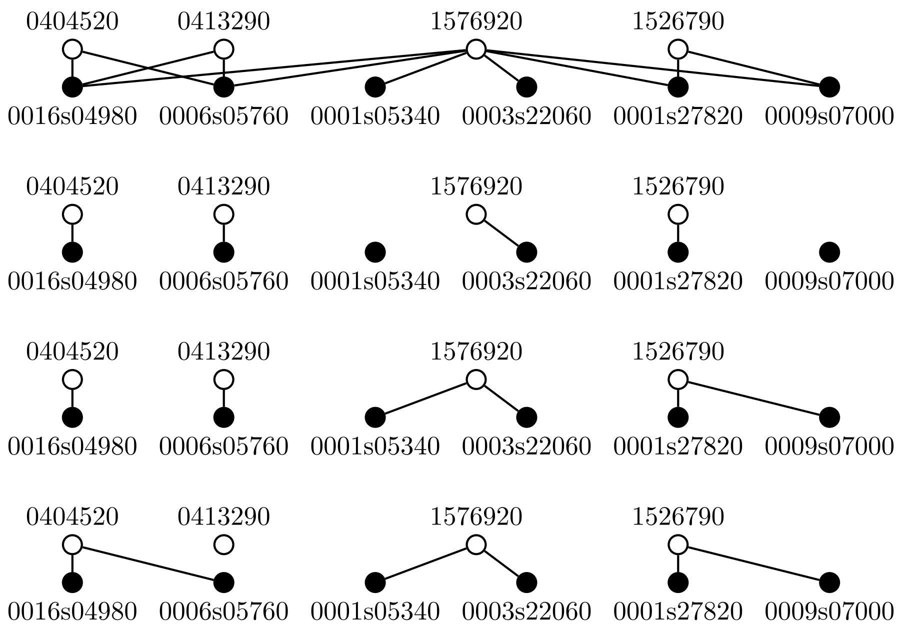

Figure 1: An example instance of OAD. The first row shows a connected component of the graph for *Ricinus communis* and *Populus trichocarpa*; white vertices represent ricinus genes, black vertices represent poplar genes, and (for simplicity) edges have unit weight. The second row shows an optimal solution to standard matching. The third row shows an optimal solution with duplication bonus $d = 0$. The fourth row shows an optimal solution with duplication bonus $d > 0$. The vertex names are the identifiers from the corresponding data sets in the CoGe database (`https://genomevolution.org/coge`).

ters: (i) one white and one black vertex connected by an edge, (ii) a path of length two where the middle vertex is white, and (iii) triangles with exactly one white vertex. Observe that in particular, we exclude the possibility of having an ortholog group in which the white vertex is considered dissimilar to one of the two black vertices. In a similar way as Orthology Assignment with Duplications, we can now formally define the Orthology Clustering with Duplications problem.

Orthology Clustering with Duplications (OCD)
Instance: An undirected graph $G = (W \uplus B, E)$, an edge-weight function $s\colon E \to \mathbb{Q}^+$.
Task: Find a partition $(C_1, \ldots, C_\ell, I)$ of $W \uplus B$ into ortholog clusters $C_1, \ldots, C_\ell$ and isolated vertices I such that $\text{val}(C_1, \ldots, C_\ell, I) := \sum_{i=1}^{\ell} \sum_{a \subseteq C_i} s(a)$ is maximized.

Note that the duplication bonus d used in the definition of Orthology Assignment with Duplications is not necessary in this model, as it can be incorporated in the input graph, by adding d to every edge weight for all edges of G connecting pairs of black vertices having at least one common neighbor—

if such an edge does not exist in G, then we may create it and assign it the weight d.

We consider in this paper only simple undirected graphs $G = (V, E)$ where the vertex set V has two disjoint subsets, a set W of white vertices and a set B of black vertices. We use $n := |W| + |B|$ to denote the number of vertices and $m := |E|$ to denote the number of edges in the input graph. For a vertex v in G, $N_G(v)$ denotes the set of neighbors of v in G.

Our analysis of OCD is in the context of parameterized complexity, where we analyze the running time of algorithms for NP-hard problems not only with respect to the input size, but also secondary parameters. Herein we try to encapsulate the combinatorial explosion in a function of the parameter. See a recent textbook [22] for more context and methodology.

3. Polynomial-Time Algorithms for Finding Orthologs in Two Genomes

In this section, we provide polynomial-time algorithms for Orthology Assignment with Duplications. First, we consider the (simpler) case in which there there is no duplication bonus, or there is a duplication penalty.

Theorem 1. Orthology Assignment with Duplications *can be solved in $O(\sqrt{n} \cdot m)$ time if $d \leq 0$.*

Proof. Construct a graph G' from the input graph G as follows. First add all black vertices of G to G'. Then replace each white vertex w by two white copies w_1, w_2 such that $N_G(w) = N_{G'}(w_1) = N_{G'}(w_2)$. The edge weights between w_1 and $N_G(w)$ are the same as the edge weights between w and $N_G(w)$, that is, for each $b \in N_G(w)$, $s(\{w_1, b\}) := s(\{w, b\})$. For w_2, we add d to these edge weights, that is, $s(\{w_2, b\}) := s(\{w, b\}) + d$ (recall that $d \leq 0$).

We will compute an optimal orthology assignment by computing a maximum-weight matching in G'. To this end, we now show that

> G has an orthology assignment with duplications of value at least ℓ
> if and only if G' has a matching of weight at least ℓ.

First, let A be an orthology assignment with duplications in G of value at least ℓ. Construct an edge set M of G' by adding for each vertex $w \in W$ the following edges. If w is incident with two edges $\{w, b_1\}$ and $\{w, b_2\}$ of A, then add $\{w_1, b_1\}$ and $\{w_2, b_2\}$ to M. If w is incident with exactly one edge $\{w, b\}$ of A then add $\{w_1, b_1\}$ to M. The edge set M is independent and the sum of the edge weights is exactly the sum of the weights of A plus d times the number of white vertices that are incident with two edges of A. Hence, the weight of the matching is the same as the value of A, as claimed.

Conversely, let M be a matching of weight at least ℓ in G'. Without loss of generality, let M be a maximum-weight matching. Consider the set A of edges of G that contains for each vertex $w \in W$ and each vertex $b \in B$, the edge $\{w, b\}$ if and only if either $\{w_1, b\} \in M$ or $\{w_2, b\} \in M$. Then, each black vertex is incident with at most one edge in A and each white vertex is incident with at most two edges in A. Hence, A is an orthology assignment with duplications. Moreover, since M has maximum weight, M contains an edge incident with w_2 only if it contains an edge incident with w_1. Thus, for each white vertex $w \in W$, there is exactly one edge incident with w in A if and only if M contains an edge that is incident with w_1 and M does not contain an edge incident with w_2. Then, the contribution of each w to A is exactly the same as the contribution of the edges that are incident with w_1 and w_2 to M: If M has only an edge incident with w_1, then the weights of the two edges in M and A are the same and w does not contribute to the second part of the value sum. Otherwise, the contribution of w to A is $s(w, b_1) + s(w, b_2) + d = s(w_1, b_1) + s(w_2, b_2)$.

The proof of the second direction of the equivalence describes a linear-time algorithm for constructing an assignment from a matching in G'. Hence, we can find an optimal assignment by first computing the graph G', then computing a maximum-weight matching in G' and then computing an assignment from this matching. The overall running time is dominated by the time needed to compute the matching. Since G' has $O(n)$ vertices and $O(m)$ edges, this step can be performed in $O(\sqrt{n} \cdot m)$ time [23]. □

We now turn to the case where $d > 0$.

Theorem 2. Orthology Assignment with Duplications *can be solved in $O(\sqrt{n} \cdot m)$ time if $d > 0$.*

Proof. Construct a graph G' from the input graph G as follows. First, add all black vertices of G to G'. Then, replace each white vertex w by two copies w_1, w_2 such that, initially $N_G(w) = N_{G'}(w_1) = N_{G'}(w_2)$. The weights of these edges are obtained from the old ones by adding a value of d each time, that is, $s(\{w_1, b\}) := s(\{w, b\}) + d$ and $s(\{w_2, b\}) := s(\{w, b\}) + d$. Further, for each pair w_1, w_2 of new white vertices that correspond to the same white vertex in W, add the edge $\{w_1, w_2\}$ to G' and assign a weight of d to it.

We will compute an optimal orthology assignment by computing a maximum-weight matching in G'. To this end, we now show that

> G has an orthology assignment with duplications of value at least ℓ
> if and only if G' has a matching of weight at least $\ell + d \cdot |W|$.

First, let A be an orthology assignment for G of value at least ℓ. Construct a matching M as follows. Consider each $w \in W$. If w is incident with two edges $\{w, b_1\}$ and $\{w, b_2\}$ in A, then add the edges $\{w_1, b_1\}$ and $\{w_2, b_2\}$ to M. If w is incident with exactly one edge $\{w, b\}$ of A, then add the edge $\{w_1, b\}$ to M. If w is not incident with any edge of A, then add the edge $\{w_1, w_2\}$ to M. In order to compare the value of A with the weight of M, we now consider each white vertex of G separately.

Case 1: w is not incident with any edge of A. Then, the contribution of w to $\mathrm{val}(A)$ is zero. By construction, M contains the edge $\{w_1, w_2\}$. Thus, the edges incident with w_1 and w_2 contribute exactly d to the weight of M.

Case 2: w is incident with exactly one edge $\{w, b\}$ of A. Then, the contribution of w to $\text{val}(A)$ is $s(\{w, b\})$. Due to construction of M, only w_1 is incident with an edge of M. The weight of this edge is $s(\{w, b\}) + d$.

Case 3: w is incident with two edges $\{w, b_1\}$ and $\{w, b_2\}$ of A. Then, the contribution of w to $\text{val}(A)$ is $s(\{w, b_1\}) + s(\{w, b_2\}) + d$. Due to construction of M, w_1 and w_2 are incident with edges of M, one having b_1 as endpoint and one having b_2 as endpoint. The weights of these edges are $s(\{w, b_1\}) + d$ and $s(\{w, b_2\}) + d$, respectively.

In each case, the contribution of w to $\text{val}(A)$ is d less than the contribution of w_1 and w_2 to the weight of M. Thus, the weight of M is at least $\text{val}(A)+d\cdot|W|$, as claimed.

Conversely, let M be a matching of weight at least $\ell+d\cdot|W|$ in G'. Without loss of generality, assume that M is a maximum-weight matching. Consider the set A of edges of G that contains for each vertex $w \in W$ and each vertex $b \in B$, the edge $\{w, b\}$ if and only if either $\{w_1, b\} \in M$ or $\{w_2, b\} \in M$. We compare again the weight of M with the value of A. We do this by grouping the contributions of the white vertices in G' in the pairs w_1, w_2. First, observe that, since M is a maximum-weight matching, at least one of w_1 or w_2 is incident with an edge in M; otherwise, adding $\{w_1, w_2\}$ to M gives a matching with higher weight. Moreover, we can assume without loss of generality, that it is w_1 that is always incident with an edge of M. Thus, it is sufficient to consider the following three cases.

Case 1: only w_1 is incident with an edge $\{w_1, b\}$ of M. Then, the contribution of w_1 to the weight of M is $s(\{w, b\}) + d$. By construction, A contains the edge $\{w, b\}$ and no other edge incident with w. Thus, the contribution of w to $\text{val}(A)$ is $s(\{w, b\})$ and therefore d less than the contribution of w_1 and w_2 to M.

Case 2: M contains $\{w_1, w_2\}$. Then, the contribution of w_1 and w_2 to M is d. By construction, w is not incident with any edge of A. Thus, the contribution of w to $\text{val}(A)$ is zero and therefore d less than the contribution of w_1 and w_2 to M.

Case 3: M contains $\{w_1, b_1\}$ and $\{w_2, b_2\}$. Then, the contribution of w_1 and w_2 to M is $s(\{w, b_1\})+d+s(\{w, b_2\})+d$. By construction, w is incident with two edges of A: the edge $\{w, b_1\}$ and the edge $\{w, b_2\}$. Thus, the contribution of w to $\text{val}(A)$ is $s(\{w, b_1\}) + s(\{w, b_2\}) + d$ and therefore it is d less than the contribution of w_1 and w_2 to M.

In each case, the contribution of w to the value of A is exactly d less than the contribution of w_1 and w_2 to the weight of M. Thus, the value of A is at least $\ell + d \cdot |W| - d \cdot |W| = \ell$, as claimed.

Thus, a maximum-weight matching in G' directly corresponds to an optimal assignment in G. Moreover, the second part of the proof gives a linear-time algorithm for computing such an assignment from the maximum-weight matching. Hence, we can find an optimal assignment by first computing the graph G', then computing a maximum-weight matching in G' and then computing an assignment from this matching. The overall running time is dominated by the time needed to compute the matching. Since G' has $O(n)$ vertices and $O(m)$ edges, this step can be performed in $O(\sqrt{n} \cdot m)$ time [23]. □

Finally, let us remark that the problem of computing a maximum-weight matching in a bipartite graph can be easily reduced to Orthology Assignment with Duplications with $d = 0$: Pick an arbitrary part of the vertex bipartition to be the white part of the Orthology Assignment with Duplications instance and add for each vertex v of this part a further vertex that has only v as neighbor and assign a sufficiently large weight to this edge. Then, any assignment will include these new edges and further edges of the assignment directly correspond to a maximum-weight matching of the original bipartite graph. A similar construction works if $d > 0$. Hence, any faster algorithm for Orthology Assignment with Duplications would directly imply a faster algorithm for computing maximum-weight matchings.

4. Including Similarities in the Duplicated Genome Makes it Hard

We now consider the more general problem where the score of an orthology group may also take information between genes of the duplicated genome into account. Unfortunately, this extension of the model results in a computationally hard problem. The reduction we use to prove NP-hardness is similar to a known NP-hardness reduction for the Cluster Editing problem [24].

Theorem 3. Orthology Clustering with Duplications *is NP-hard, even when the input graph has maximum degree six and weights are all unitary.*

Proof. The reduction is from the 3-SAT problem where we have a Boolean formula ϕ in conjunctive normal form and clauses of size exactly three and we ask whether ϕ is satisfiable. Let N be the number of variables in ϕ and M the number of its clauses. Denote the maximum number of occurrences of a variable in any clause by ℓ. (Without loss of generality, we assume that each variable occurs only once in each clause.) We show how to construct in polynomial time a white/black-colored graph G which admits a set of vertex-disjoint ortholog clusters containing at least $3\ell N$ edges if and only if ϕ is satisfiable. This directly implies the NP-hardness of Orthology Clustering with Duplications.

Let us construct G, which is the empty graph initially. First, add N cycles, each with 2ℓ black vertices, corresponding to the variables in ϕ. For each cycle, label the edges in an arbitrary maximum matching by *positive* and the remaining edges by *negative.* This labeling is used only in the construction and will not be part of the final instance. Next, for each clause, add a white vertex v and attach it to the cycles of the variables it contains as follows. For each positive literal x, make v adjacent to the two endpoints of a positive edge of the cycle of x. For each negative literal $\neg x$, make v adjacent to the endpoints of a negative edge of the cycle of x. For each of these connections, choose an edge whose endpoints are not yet connected to any clause vertex. Note that this is always possible since each cycle has length 2ℓ. Finally, introduce $\ell N - M$ white *dummy* vertices that are connected to each vertex in each cycle. (Note that $M \leq \ell N/3$, so this is possible.) We assign each edge weight one, meaning that the value of a clustering is simply the number of edges contained in its clusters. This concludes the construction of G. As indicated above, we ask for an ortholog clustering of at least $3\ell N$ edges.

Now assume that ϕ is satisfiable and fix an arbitrary satisfying assignment of the variables. Hence, for each clause there is a literal satisfying this clause. To construct a clustering for G, for each clause C, we pick as a ortholog cluster a triangle in G which consists of the white vertex corresponding to the clause and the endpoints of an edge in a variable cycle. The variable is chosen such that its literal in C is satisfied, that is, if the literal is positive, then the edge in the triangle is positive and vice versa. Note that we can indeed pick these clusters, because, first, by the way we connected the clause vertices and variable cycles, no positive (resp. negative) edge blocks another positive (resp. negative) edge from being taken into a cluster, and, second, because choosing the clusters in this way, we consistently choose only positive or only negative edges from each variable cycle. Hence, we obtain M clusters with three edges each. Finally, for each variable cycle, we complete the set of cluster edges in it to a perfect matching, by making all remaining matching edges into a cluster with an arbitrary dummy vertex. In total, we obtain $M + \ell N - M = \ell N$ ortholog clusters, each with three edges. That is, the ORTHOLOGY CLUSTERING WITH DUPLICATIONS instance has a solution.

Now assume that G has a set of ortholog clusters with at least $3\ell N$ edges overall. Observe that G contains exactly ℓN white vertices and thus, at least ℓN edges inside the ortholog clusters are between black vertices. Since in each of the N variable cycles there are at most ℓ edges in clusters, and these are the only "black" edges, the clusters induce perfect matchings in each variable cycle. Thus, for each cycle, the edges correspond to a valid assignment of truth values to the variables: Assign each variable x in ϕ the value true if the cluster edges from the corresponding cycle are positive and assign it false, otherwise. Since there are only $\ell N - M$ white dummy vertices, all M clause vertices are in ortholog clusters. By the way in which we connected the clause vertices to the variable cycles, this means that each clause is satisfied by ϕ. □

The NP-hardness of OCD motivates fixed-parameter algorithms for this problem. Observe that OCD can be viewed as an edge deletion problem: delete a minimum-weight set of edges such that only the edges of the ortholog clusters C_i remain. The standard parameter for such problems is the number of edge deletions k to obtain a solution [25, 26]. It is easy to obtain fixed-parameter tractability for OCD and parameter k as the following shows.

Proposition 1. *Let $(G = (V, E), s)$ be an instance of* OCD *such that no connected component is an ortholog cluster, let $(C_1, \dots, C_\ell, I)$ be an ortholog clustering of G, and let k denote the number of edges not contained in any ortholog cluster C_i. Then, $|E| \leq 7k$.*

Proof. In an instance as described above, every ortholog cluster is incident with at least one edge deletion. Thus, the number of ortholog clusters is at most $2k$. Since an ortholog cluster has at most three edges, this implies that such a graph has at most $7k$ edges: at most $6k$ edges in the ortholog clusters and at most k deleted edges. $\square$

The above proposition shows that k is not a small parameter. In order to obtain more efficient fixed-parameter algorithms, we consider two smaller parameters. The first parameter is as follows.

Definition 2 (Parameter k_B)**.** *Given an instance of* OCD*, by k_B we denote the number of edge deletions necessary to destroy all paths on three vertices that start and end with a white vertex.*

Clearly, such paths cannot be contained completely in any ortholog cluster C_i. Hence, $k_B \leq k$. The second parameter is defined below.

Definition 3 (Parameter k_W)**.** *Given an instance of* OCD*, by k_W we denote the number of edge deletions necessary to destroy all subgraphs containing a white vertex with degree three.*

In an ortholog cluster, every white vertex has degree at most two. Hence, $k_W \leq k$. A further advantage of k_B and k_W is that they can be easily computed as in fact $k_B = \sum_{b\in B} \max\{|N(b) \cap W| - 1, 0\}$ and $k_W = \sum_{w\in W} \max\{|N(w)| - 2, 0\}$ (recall that white vertices have only black neighbors). Hence, one may easily decide whether k_B or k_W are small, and thus whether they are appropriate parameters.

In the following, we say that a *reduction rule* is *correct* if the objective value of an optimal solution is the same before and after the rule is applied. Similarly, a *branching rule* is *correct* if the optimal solution of one of the recursive branches gives an optimal solution of the instance to which the rule is applied.

A fixed-parameter algorithm for k_B. We first describe a search tree algorithm for parameter k_B. The strategy of the algorithm is to first destroy all paths between white vertices.

Branching Rule 4. *If B contains a vertex b with at least two neighbors $w_1, w_2 \in W$, branch into the two cases to delete either $\{b, w_1\}$ or $\{b, w_2\}$.*

The branching rule is correct since any ortholog cluster can contain at most one of $\{b, w_1\}$ and $\{b, w_2\}$. In the recursive branches, parameter k_B is decreased by at least one. Moreover, if $k_B = 0$, the rule does not apply. Since this is the only branching rule that we apply, this implies that the resulting search tree has size $O(2^{k_B})$.

We now show that all instances for which the rule does not apply can be solved in linear time by applying two reduction rules.

Reduction Rule 5. *If G contains an edge $\{b_1, b_2\}$ such that $b_1, b_2 \in B$ do not have a common neighbor in W, then delete $\{b_1, b_2\}$.*

Reduction Rule 6. *If G contains a connected component G' that is not an ortholog cluster and has exactly one white vertex w, then compute an ortholog cluster C such that C contains w and $\sum_{a\subseteq C} s(a)$ is maximum. Delete all edges of G' that are not contained in C.*

Lemma 1. *Reduction Rules 5 and 6 are correct and can be executed exhaustively in $O(m+n)$ time.*

Proof. Reduction Rule 5 is correct since no such edge $\{b_1, b_2\}$ can be part of an ortholog cluster. It can be applied exhaustively in $O(m+n)$ time since every vertex in B has at most one neighbor in W (because Branching Rule 4 does not apply by hypothesis).

The correctness of Reduction Rule 6 is implied by two facts. First, any ortholog cluster containing a vertex from G' contains w. Second, in any optimal solution, any vertex of G' is either contained in an ortholog cluster with w or belongs to the set I of isolated vertices.

The running time can be seen as follows. Say that a connected component C of G is *amenable* if it fulfills the condition of Reduction Rule 6, that is, C is not an ortholog cluster and contains exactly one white vertex. First, in $O(m+n)$ time, we compute the amenable connected components of G. Then, for each such connected component do the following. If w has only one neighbor b in G', then $\{w, b\}$ is the optimal cluster C and we are done. Otherwise, we first find the two edges incident with w that have the largest two edge weights. The sum of these two edge weights gives the best ortholog cluster among those that do not contain an edge between the two black vertices. Thus, initialize C with the three corresponding vertices. Then, for each edge e incident with two black vertices b_1 and b_2, we check whether the endpoints of the edge are both adjacent to w. If this is the case, then compute in $O(1)$ time the sum $s(b_1, b_2)+s(b_1, w)+s(b_2, w)$. If this sum exceeds the sum of the edges in the current best ortholog cluster C, then update C accordingly.

Clearly, this procedure computes the optimal ortholog cluster C for each amenable connected component. All edges not contained in any such cluster C but in amenable connected components can be deleted in linear time. Now the overall running time follows from the fact that deleting edges in amenable connected components does not create new amenable connected components. □

Theorem 7. Orthology Clustering with Duplications *can be solved in $O(2^{k_B} \cdot (m+n))$ time.*

Proof. The algorithm first applies Branching Rule 4 as long as possible. Afterwards, in each leaf of the corresponding search tree, it applies Reduction Rules 5 and 6 exhaustively. In the remaining instance all connected components are ortholog clusters and we can compute the value of the corresponding clustering by computing the sum of the edge weights in the graph associated with the leaf of the search tree. By keeping the best clustering found in any leaf, the algorithm computes an optimal solution.

As argued above, the search tree produced by the algorithm has size $O(2^{k_B})$. Hence, the running time bound can be obtained by showing that in each search tree node, we spend $O(m+n)$ time. Clearly, we can check in $O(m+n)$ time whether Branching Rule 4 applies. By Lemma 1, Reduction Rules 5 and 6 can be applied exhaustively in $O(m+n)$ time. Finally, if these rules do not apply, then the value of the ortholog clustering is computed by summing all edge weights in $O(m + n)$ time. $\square$

A fixed-parameter algorithm for k_W. The approach is again to first destroy all subgraphs containing a white vertex of degree at least three via branching. Instances that do not contain such a vertex are solved by computing a maximum-weight matching in an auxiliary graph.

Branching Rule 8. *If B contains a vertex $w \in W$ with at least three neighbors $b_1, b_2, b_3 \in B$, branch into the three cases to delete either $\{w, b_1\}$, $\{w, b_2\}$, or $\{w, b_3\}$.*

The correctness follows from the fact that in every ortholog cluster, the white vertex has degree at most two.

Lemma 2. *Let (G, s) be an instance of* Orthology Clustering with Duplications *such that every vertex in W has degree at most two. Then, (G, s) can be solved in $O(\sqrt{n} \cdot m)$ time.*

Proof. First, apply Reduction Rule 5 exhaustively to G; this can be done in $O(m + n)$ time. Afterwards, construct an auxiliary graph G' with edge weight function s' derived from G and s as follows. The graph G' is obtained from G by adding an edge between all black vertices that have a common white neighbor. Observe that by the condition of the lemma, each white vertex has at most two neighbors. Thus, the number of added edges is $O(n)$. Then, for every edge $\{b_1, b_2\}$ between two black vertices in G' that share at least one common white neighbor, carry out the following operation. Let $w \in W$ be a common neighbor such that $s(b_1, w) + s(b_2, w)$ is maximum. Then, set $s'(b_1, b_2) := s(b_1, b_2) + s(b_1, w) + s(b_2, w)$ if $\{b_1, b_2\}$ is contained in G and $s'(b_1, b_2) := s(b_1, w) + s(b_2, w)$, otherwise. Intuitively, this means that the new weight of $\{b_1, b_2\}$ records not only the similarity between b_1 and b_2 but instead the weight of the best ortholog cluster that contains b_1 and b_2. For all other edges set $s'(a) := s(a)$. We now show that the best ortholog clustering in (G, s) corresponds to a maximum-weight matching in (G', s'). More precisely, we show that (G, s) has an ortholog clustering of value ℓ if and only if (G', s') has a matching of weight ℓ.

First, let $(C_1, \ldots, C_\ell)$ be an ortholog clustering of value ℓ. Construct a matching M in (G', s') as follows: For each cluster C_i consisting of exactly two vertices b and w add the edge $\{b, w\}$ to M. For each cluster C_i containing three vertices b_1, b_2, and w where $b_1, b_2 \in B$ add the edge $\{b_1, b_2\}$ to M. By definition, the sum of the edge weights of M in G' is at least ℓ. Moreover, M is a matching since the C_is have disjoint vertex sets.

For the converse direction, we mainly have to prove that clusters corresponding to matching edges are pairwise vertex disjoint. If a matching edge $e \in M$ is between two black vertices b_1, and b_2, then let w be the vertex such that $s(b_1, w) + s(b_2, w)$ is maximum. Construct an ortholog cluster C containing b_1, b_2, and w and the edges between them. The weight of the edges of C in G

is, by definition of s', exactly $s'(b_1, b_2)$. If a matching edge $e \in M$ is between a black and a white vertex, then construct an ortholog cluster containing the endpoints of e and u. Again by definition of s', the weight of the edge e in G' and the value of the ortholog cluster in G are the same. It remains to show that no two ortholog clusters have a vertex in common. The only possibility for a vertex to be in two ortholog clusters occurs when a white vertex w is added to a matching edge $\{b_1, b_2\}$ to produce a cluster. By the condition of the lemma, vertex w has no further neighbors in G' and thus the pair $\{b_1, b_2\}$ is the only pair that may cause an addition of w to a cluster. Thus, the produced set of vertex sets is an ortholog clustering of weight ℓ.

Observe that the converse direction of the proof also describes a linear-time algorithm to construct an optimal ortholog clustering from a maximum-weight matching in G'. The overall running time bound follows from the fact that G' can be constructed in $O(n + m)$ time and has n vertices and $O(m)$ edges. $\square$

We immediately obtain the following running time bound on the search tree algorithm by observing that Branching Rule 8 produces a search tree of size $O(3^{k_W})$.

Theorem 9. Orthology Clustering with Duplications *can be solved in $O(3^{k_W} \cdot \sqrt{n} \cdot m)$ time.*

5. Experiments

In the following, we report on our empirical findings for the Orthology Assignment with Duplications (OAD) and Orthology Clustering with Duplications (OCD) models and the algorithms developed for them in Sections 3 and 4.

Data Acquisition. We built instances based on the genomes of *Ricinus communis* [27] (castor bean), *Populus trichocarpa* [28] (western balsam poplar), and *Theobroma cacao* [29] (cacao tree). Of these three species, *Populus trichocarpa* has undergone a whole genome duplication after the speciation events. Accordingly, we built one set of instances with *Ricinus communis* as white species and one set of instances with *Theobroma cacao* as white species; *Populus trichocarpa* is the black species in both sets. In the following, we refer to the three species as *Ricinus*, *Cacao*, and *Populus*.

To build the graphs, following the methodology introduced previously [5, 9], we computed BLAST Expect (E) values [30] which measure the expected number of hits with the same alignment score that would be encountered in a random string of the same size. We add an edge between two genes if the E-value is below some threshold. We use three threshold values $t \in \{0.0, 10^{-140}, 10^{-80}\}$. Each edge in the built graphs has weight one. To keep the size of the OCD instances reasonable, we use an explicit duplication bonus d instead of creating edges between all pairs of black vertices as described in Section 1 (which is equivalent in the unweighted case).

Table 1: Some general properties of the instances for OAD and OCD. Here, n_B is the number of black vertices, n_W the number of white vertices, and m_{OAD} and m_{OCD} the number of edges in the OAD and OCD instances, respectively. By n^* we denote the order of the largest component and with $n \mid 95$ the 95th percentile of the component orders, and analogously we denote the maximum and 95th percentile values of k_B and k_W.

	Ricinus vs. *Populus*			*Cacao* vs. *Populus*		
t	10^{-80}	10^{-140}	0.0	10^{-80}	10^{-140}	0.0
n_B	24472	18079	14235	25165	18140	14287
n_W	15111	11333	9075	16963	12488	9907
n^*	1914	316	201	2261	440	256
$n \mid 95$	12	10	9	13	11	9
m_{OAD}	132446	56438	34445	227778	77887	44251
m_{OCD}	269862	109352	64162	368430	131568	73837
# comp.	7980	7102	6247	7847	7043	6152
# nontr. comp	2493	1897	1504	2698	2008	1613
# diff. comp	2141	1620	1272	2373	1806	1456
k_B^*	30351	2848	1268	65613	5957	3723
k_W^*	30608	2895	1327	65300	5946	3723
$k_B \mid 95$	20	13	8	25	15	11
$k_W \mid 95$	20	12	8	24	13	9

Implementation Details. The algorithms were implemented in Python 2.7.6 using NetworkX 1.11.[5] For OAD, we used a straightforward implementation of the algorithms described in Section 3. For OCD we implemented the fixed-parameter algorithm for parameter k_B since numerical evaluation showed that the search tree size for this parameter was usually smaller than for k_W. To improve the running time, apart from Reduction Rules 5 and 6, we implemented further reduction rules related to situations in which vertices' neighbors are subsets of other vertices' neighbors and where the optimal matches of a white vertex can be determined locally. If $k_W = 0$ during the execution of the algorithm, then we use the algorithm from Lemma 2 to solve the remaining instance.

Instance Properties. Some general instance properties and statistics about the results for OAD are shown in Table 1. As the table shows, the graphs decompose into a large number of smaller connected components, most of which are trivial, that is, they consist of one white vertex and one or two black vertices. The largest connected components contain between 201 and 1914 vertices. The majority of the nontrivial components have at most 10 vertices. Similarly, for the majority of nontrivial components, the two parameters k_B and k_W have moderate values. Note also that the size of the instances, in terms of the number of edges, increases roughly twofold between each increment of the threshold value. Each edge between two black genes represents a possible paralog. These possible paralogs are taken into account in the OCD instances and not in the OAD instances. They make up roughly half of the edges in the OCD instances.

Solution Statistics for OAD *and* OCD*.* We computed the solutions for OAD and OCD for the instances described above. For OCD, not all connected com-

[5]Source code and results are available at `http://fpt.akt.tu-berlin.de/oad`.

Table 2: Number and type of the clusters output for OAD and OCD. Here, d is the duplication bonus, C is the total number of clusters, B is the number of covered black genes, and $\%\{P_2, P_3, K_3\}$ is the number of the clusters of the corresponding type.

	OAD					OCD				
d	C	B	$\%P_2$	$\%P_3$	$\%K_3$	C	B	$\%P_2$	$\%P_3$	$\%K_3$
Ricinus vs. *Populus*, $t = 10^{-80}$										
-1.1	14811	14811	100.00	0.00	0.00	10274	15212	51.94	0.00	48.06
-0.5	14811	22724	46.57	6.41	47.01	9571	15338	39.75	1.44	58.81
0.0	13826	22724	35.64	7.83	56.53	9547	15337	39.35	1.70	58.95
1.0	13293	22724	29.05	9.10	61.85	9539	15337	39.22	1.82	58.96
Ricinus vs. *Populus*, $t = 10^{-140}$										
-1.1	11085	11085	100.00	0.00	0.00	8879	12994	53.65	0.00	46.35
-0.5	11085	16756	48.84	4.97	46.19	8356	13082	43.44	1.26	55.30
0.0	10547	16756	41.13	5.87	53.00	8332	13082	42.99	1.55	55.46
1.0	10216	16756	35.98	6.60	57.42	8319	13080	42.77	1.71	55.52
Ricinus vs. *Populus*, $t = 0.0$										
-1.1	8873	8873	100.00	0.00	0.00	7784	11319	54.59	0.00	45.41
-0.5	8873	13228	50.92	3.87	45.22	7362	11379	45.44	0.91	53.65
0.0	8498	13228	44.34	4.28	51.38	7337	11378	44.92	1.28	53.80
1.0	8258	13228	39.82	5.16	55.03	7323	11379	44.61	1.52	53.87
Cacao vs. *Populus*, $t = 10^{-80}$										
-1.1	16154	16154	100.00	0.00	0.00	10373	15195	53.51	0.00	46.49
-0.5	16154	24048	51.13	7.69	41.17	9581	15349	39.80	2.10	58.10
0.0	14624	24048	35.56	10.84	53.60	9514	15345	38.71	2.86	58.43
1.0	13913	24048	27.15	12.86	59.99	9498	15344	38.45	3.11	58.44
Cacao vs. *Populus*, $t = 10^{-140}$										
-1.1	11887	11887	100.00	0.00	0.00	9040	12996	56.24	0.00	43.76
-0.5	11887	17371	53.87	4.59	41.54	8371	13075	43.81	1.22	54.98
0.0	10960	17371	41.51	6.51	51.98	8341	13073	43.27	1.61	55.13
1.0	10514	17371	34.78	8.07	57.14	8325	13071	42.99	1.84	55.17
Cacao vs. *Populus*, $t = 0.0$										
-1.1	9469	9469	100.00	0.00	0.00	7883	11301	56.64	0.00	43.36
-0.5	9469	13708	55.23	3.89	40.88	7334	11357	45.15	1.04	53.82
0.0	8802	13708	44.26	4.79	50.94	7303	11352	44.56	1.47	53.98
1.0	8502	13708	38.77	5.80	55.43	7293	11352	44.34	1.70	53.96

ponents could be solved within reasonable time as our algorithms have exponential running times in the worst case. Hence, we set a timeout of 60 seconds for each connected component. With this setting, the connected components solved by OCD contained altogether between a minimum of 63% of the vertices (attained for *Cacao* and $t = 10^{-80}$) and a maximum of 87% (for *Ricinus* and $t = 0.0$), and the total running time for an instance of OCD ranged between 2.1 and 6.7 hours. In contrast, all OAD instances could be solved in reasonable time, with the largest instance taking 2 minutes.

Table 2 shows solution statistics for OAD and OCD for different values of the duplication bonus d and the distribution of the ortholog clusters into P_2-*clusters* (single edges), P_3-*clusters* (a white vertex with two black neighbors that are not adjacent in the OCD instance), and K_3-*clusters* (a white vertex and two black vertices, pairwise adjacent in the OCD instance). Observe that,

Table 3: Number and type of the clusters output for OAD and OCD restricted to components that were solved by both methods. Here, d is the duplication bonus, C is the total number of clusters, B is the number of covered black genes, and $\%\{P_2, P_3, K_3\}$ is the number of the clusters of the corresponding type.

	OAD					OCD				
d	C	B	$\%P_2$	$\%P_3$	$\%K_3$	C	B	$\%P_2$	$\%P_3$	$\%K_3$
					Ricinus vs. *Populus*, $t = 10^{-80}$					
-1.1	10274	10274	100.00	0.00	0.00	10274	15212	51.94	0.00	48.06
-0.5	10274	15345	50.64	3.66	45.70	9571	15338	39.75	1.44	58.81
0.0	9819	15345	43.72	4.25	52.03	9547	15337	39.35	1.70	58.95
1.0	9527	15345	38.93	4.66	56.41	9539	15337	39.22	1.82	58.96
					Ricinus vs. *Populus*, $t = 10^{-140}$					
-1.1	8879	8879	100.00	0.00	0.00	8879	12994	53.65	0.00	46.35
-0.5	8879	13082	52.66	2.95	44.39	8356	13082	43.44	1.26	55.30
0.0	8534	13082	46.71	3.48	49.81	8332	13082	42.99	1.55	55.46
1.0	8304	13082	42.46	3.96	53.58	8319	13080	42.77	1.71	55.52
					Ricinus vs. *Populus*, $t = 0.0$					
-1.1	7784	7784	100.00	0.00	0.00	7784	11319	54.59	0.00	45.41
-0.5	7784	11382	53.78	2.62	43.60	7362	11379	45.44	0.91	53.65
0.0	7505	11382	48.34	2.97	48.69	7337	11378	44.92	1.28	53.80
1.0	7312	11382	44.34	3.62	52.04	7323	11379	44.61	1.52	53.87
					Cacao vs. *Populus*, $t = 10^{-80}$					
-1.1	10373	10373	100.00	0.00	0.00	10373	15195	53.51	0.00	46.49
-0.5	10373	15352	52.00	4.85	43.15	9581	15349	39.80	2.10	58.10
0.0	9821	15352	43.68	5.85	50.46	9514	15345	38.71	2.86	58.43
1.0	9485	15352	38.14	6.49	55.36	9498	15344	38.45	3.11	58.44
					Cacao vs. *Populus*, $t = 10^{-140}$					
1.1	9040	9040	100.00	0.00	0.00	9040	12996	56.24	0.00	43.76
-0.5	9040	13080	55.31	2.75	41.94	8371	13075	43.81	1.22	54.98
0.0	8580	13080	47.55	3.52	48.93	8341	13073	43.27	1.61	55.13
1.0	8318	13080	42.75	4.24	53.01	8325	13071	42.99	1.84	55.17
					Cacao vs. *Populus*, $t = 0.0$					
-1.1	7883	7883	100.00	0.00	0.00	7883	11301	56.64	0.00	43.36
-0.5	7883	11360	55.89	2.87	41.24	7334	11357	45.15	1.04	53.82
0.0	7486	11360	48.25	3.09	48.66	7303	11352	44.56	1.47	53.98
1.0	7291	11360	44.19	3.69	52.12	7293	11352	44.34	1.70	53.96

while the OAD instances contain no edges between black vertices, we use these edges to compute the cluster distribution.

Using OAD, the ratio of white genes covered by clusters ranges between 82 % and 98 % over all instances, threshold values, and duplication bonuses. The coverage of black genes ranges between 92 % and 95 % for duplication bonuses between -0.5 and 1.0, and around 60 % if the bonus is -1.1. Indeed, for OAD with $d = -1.1$ we obtain the same solutions that we would obtain by computing a maximum matching between the white and black genes. In contrast, for OCD with $d = -1.1$ we see a sharp divide into clusters of size two and K_3-clusters. OCD failed to solve larger components; Table 3 compares OAD and OCD on those components that were solved by both variants. Extrapolating from these data, OCD covers a similar ratio of black and white genes as OAD, except for the case of black genes and duplication bonus -1.1: The coverage of black genes for OCD with duplication bonus -1.1 does not differ substantially from the one for other duplication bonuses.

Generally, we see that higher duplication bonus leads to a lower number of clusters meaning less coverage for the set of white genes. Observe that, somewhat counterintuitively, the percentage of K_3-clusters is sometimes higher for OAD although OAD ignores these edges. The reason for this is simply that OCD failed to solve the larger components which contain more potential paralogs: The comparison of the OCD results to the OAD results restricted to the components that were solved by OCD in Table 3 shows that OCD always identifies more K_3-clusters. Notably, the number of P_3-clusters identified by the OCD model is at most half the one for OAD.

Similarly to the number of K_3-clusters, it may seem that the number of covered black genes is much lower for OCD, but Table 3 reveals that the coverage is only marginally smaller when restricted to the components both algorithms could solve.

In summary, we see that OAD and OCD cover in ortholog clusters large ratios of both black and white genes, regardless of the threshold t. The coverage between the two methods is similar, barring computational barriers, except for duplication bonus -1.1, where it is significantly lower for OAD. In contrast to OAD, OCD produces roughly half the number of P_3-clusters and slightly more K_3-clusters.

Similarity of Predicted Paralogs for OAD *and* OCD. The solution statistics show that both OAD and OCD identify large numbers of supposed orthologs. We now evaluate the biological quality of these results using three tests.

Recall that the clusters of size three for OAD and OCD are in fact predictions of pairs of paralogs in the duplicated genome. In our case, this is the *Populus* genome. It is expected that paralogs have high sequence similarity. In the first test, we thus evaluate the Blast E-values of the predicted paralogs.

Moreover, we expect that the sequence similarity for the predicted paralogs of *Populus* is higher than both sequence similarities between the *Populus* genes and the *Ricinus* or *Cacao* genes. In other words, we expect that the gene tree of the three sequences in the cluster agrees with the species tree of the two species. We check whether this is indeed the case in the second test, a standard test in comparative genomics.

As the third and final test, for each pair of predicted paralogs, we retrieved and compared their chromosome number. Paralogs that arise due to the genome duplication event should be located on different chromosomes. Hence, we computed the percentage of paralogs that are located on different chromosomes. We expect that a higher percentage indicates a higher percentage of correctly assigned paralogs.

Among the matched genes, in general, smaller threshold values lead to larger percentages for correct gene trees regardless of which of the two methods we use. For OAD the number of output clusters with correct gene trees range between at least 50 % and at least 90 % with increases of circa 20 % per decrement of

Table 4: Average sequence similarity for the predicted paralogs and percentage of clusters for which the gene tree matches the sequence tree using the OAD model. Here d denotes the duplication bonus and $e \mid x$ the BLAST E-value at the xth percentile, % cgt denotes the percentage of size-three clusters with correct gene trees, % dc denotes the percentage size-three clusters whose predicted paralogs are located on different chromosomes of *Populus*.

d	$e \mid 75$	$e \mid 50$	$e \mid 25$	% cgt	% dc
		Ricinus vs. *Populus*, $t = 10^{-80}$			
−1.1	—	—	—	—	—
−0.5	3.35e-112	2.91e-161	0.00e+00	61.03	83.89
0.0	2.48e-111	5.32e-159	0.00e+00	60.46	83.56
1.0	4.43e-110	3.43e-157	0.00e+00	59.66	83.52
		Ricinus vs. *Populus*, $t = 10^{-140}$			
−1.1	—	—	—	—	—
−0.5	4.67e-177	0.00e+00	0.00e+00	77.41	85.15
0.0	3.85e-176	0.00e+00	0.00e+00	76.63	84.81
1.0	4.75e-175	0.00e+00	0.00e+00	75.98	84.72
		Ricinus vs. *Populus*, $t = 0$			
−1.1	—	—	—	—	—
−0.5	0.00e+00	0.00e+00	0.00e+00	93.25	85.74
0.0	0.00e+00	0.00e+00	0.00e+00	93.28	86.03
1.0	0.00e+00	0.00e+00	0.00e+00	92.64	85.77
		Cacao vs. *Populus*, $t = 10^{-80}$			
−1.1	—	—	—	—	—
−0.5	7.13e-105	9.71e-154	0.00e+00	58.42	83.80
0.0	2.96e-103	4.88e-150	0.00e+00	56.92	82.49
1.0	2.18e-101	8.21e-148	0.00e+00	55.93	82.47
		Cacao vs. *Populus*, $t = 10^{-140}$			
−1.1	—	—	—	—	—
−0.5	7.25e-177	0.00e+00	0.00e+00	76.93	84.74
0.0	6.60e-174	0.00e+00	0.00e+00	75.35	83.23
1.0	1.53e-170	0.00e+00	0.00e+00	73.79	83.29
		Cacao vs. *Populus*, $t = 0$			
−1.1	—	—	—	—	—
−0.5	0.00e+00	0.00e+00	0.00e+00	92.57	85.77
0.0	0.00e+00	0.00e+00	0.00e+00	92.38	85.32
1.0	0.00e+00	0.00e+00	0.00e+00	91.55	84.67

the threshold value. Here, we usually observe that between 50 % and 75 % of the presumed paralogs have alignment E-values of 0.0. For OCD these numbers increase substantially and we usually observe that more than 75 % of the presumed paralogs have perfect E-values, and the percentage of clusters with correct gene trees is never below 70 %. Surprisingly, increasing the duplication bonus decreases the alignment scores throughout, and also the percentage of correct gene trees decreases. For sake of completeness, Tables 6 and 7 in the appendix contain the results restricted to the components that were solved by both methods.

The percentage of clusters in which the black genes reside on different chromosomes ranges from 83% to 86% for OAD and from 88% to 90% for OCD. However, when restricting to components solved by both algorithms, the numbers for OAD become similar to the ones for OCD, with only a small tendency

Table 5: Average sequence similarity for the predicted paralogs and percentage of clusters for which the gene tree matches the sequence tree using the OCD model. Here d denotes the duplication bonus and $e \mid x$ the BLAST E-value at the xth percentile, % cgt denotes the percentage of size-three clusters with correct gene trees, % dc denotes the percentage size-three clusters whose predicted paralogs are located on different chromosomes of *Populus*.

d	$e \mid 75$	$e \mid 50$	$e \mid 25$	% cgt	% dc
Ricinus vs. *Populus*, $t = 10^{-80}$					
-1.1	3.94e-134	0.00e+00	0.00e+00	75.17	90.28
-0.5	1.16e-129	0.00e+00	0.00e+00	71.87	88.56
0.0	3.07e-129	0.00e+00	0.00e+00	71.61	88.50
1.0	3.52e-129	0.00e+00	0.00e+00	71.51	88.51
Ricinus vs. *Populus*, $t = 10^{-140}$					
-1.1	0.00e+00	0.00e+00	0.00e+00	88.80	89.99
-0.5	0.00e+00	0.00e+00	0.00e+00	85.65	88.68
0.0	0.00e+00	0.00e+00	0.00e+00	85.22	88.72
1.0	0.00e+00	0.00e+00	0.00e+00	85.02	88.66
Ricinus vs. *Populus*, $t = 0$					
-1.1	0.00e+00	0.00e+00	0.00e+00	100.00	89.70
-0.5	0.00e+00	0.00e+00	0.00e+00	98.53	88.35
0.0	0.00e+00	0.00e+00	0.00e+00	98.07	88.29
1.0	0.00e+00	0.00e+00	0.00e+00	97.71	88.24
Cacao vs. *Populus*, $t = 10^{-80}$					
-1.1	2.60e-134	0.00e+00	0.00e+00	75.01	90.40
-0.5	8.48e-128	5.87e-180	0.00e+00	70.96	88.56
0.0	6.78e-126	1.98e-178	0.00e+00	70.11	88.44
1.0	2.34e-125	3.90e-178	0.00e+00	69.88	88.33
Cacao vs. *Populus*, $t = 10^{-140}$					
-1.1	0.00e+00	0.00e+00	0.00e+00	88.60	89.26
-0.5	0.00e+00	0.00e+00	0.00e+00	85.71	87.80
0.0	0.00e+00	0.00e+00	0.00e+00	85.23	87.81
1.0	0.00e+00	0.00e+00	0.00e+00	84.91	87.86
Cacao vs. *Populus*, $t = 0$					
-1.1	0.00e+00	0.00e+00	0.00e+00	100.00	90.14
-0.5	0.00e+00	0.00e+00	0.00e+00	98.36	88.44
0.0	0.00e+00	0.00e+00	0.00e+00	97.70	88.44
1.0	0.00e+00	0.00e+00	0.00e+00	97.31	88.40

for improvement in OCD (see Tables 6 and 7 in the appendix).

In summary, all three tests give encouraging results, showing that our methods may yield biologically significant predictions of orthologs. It is especially noteworthy that, despite the fact that our methods do not explicitly take gene positions into account, circa 90 % of the predicted paralogs reside on different chromosomes and hence pass the chromosome-based paralogy test. Comparing OAD and OCD, while for the sequence similarity-based tests the results are always better for OCD than for OAD, the results for the chromosome test are less clear.

It seems that using $d = -0.5$ for OAD and $d = -1.1$ for OCD are reasonable settings: the first obtains the best coverage in terms of numbers of genes in the clusters while having the best quality among the OAD runs. The second setting gives the best overall quality. Since the coverage for OCD is generally much lower than for OAD, due to the fact that the time limit is exceeded on many larger connected components, it is advisable to use OCD only if the cluster quality is much more important than overall coverage.

Preliminary Comparison with InParanoid. In order to get a preliminary point of comparison we chose InParanoid 4.1 [4, 17–20], one of the well-established graph-based methods for identifying ortholog clusters, chosen for ease of use and because it is a balanced approach, yielding consistently good performance over a variety of benchmarks [31]. We used it to output ortholog clusters with paralogs on the *Ricinus* and *Populus* instances. With the standard parameter settings, InParanoid identifies 95 ortholog clusters. Of these clusters 19 have size three, all of them contain two *Populus* genes and one *Ricinus* gene ; 9 clusters have size more than three. Hence, InParanoid with standard settings does not produce a clustering with full resolution as our method does. The quality of the size-three clusters is perfect according to our tests: in all of them, the gene tree fits the species tree and for all of them the *Populus* genes are located on different chromosomes. Possibly, the high quality of the size-three clusters can be explained with the strict similarity thresholds that are applied for considering any pair of genes as homologs in the standard setting of InParanoid. In any case, even the most strict parameter setting for OAD and OCD yields a much higher number of clusters, at the cost of cluster quality. In the future, we plan to investigate more thoroughly the effect of different parameter settings for InParanoid for these instances.

Conclusions

We presented two models for identifying orthologs and paralogs in the presence of whole genome duplication. Experiments yield encouraging results, indicating biological relevance of the obtained ortholog groups. While the more sophisticated model gives better results it is, due to its NP-hardness, algorithmically more challenging and current approaches cannot solve all realistic instances. Consequently, further algorithmic improvements have to be made in order to fully estimate the biological quality of its results.

While our goal here was to first evaluate an approach that does not directly take into account the positions of orthologous genes, the chromosome test shows that some (very coarse) positional information is recovered by our model. More precise tests based on positional data are needed, to determine which of these data may be helpful to incorporate into the model and yield more relevant orthologs. It would be worthwhile to study a straightforward extension of our models in which such positional data is included when computing similarity scores between genes. More generally, it would be interesting to see whether introducing edge weights leads to improved results, but in this context, an appropriate way of determining the duplication bonus becomes critical. More complicated combinatorial models that incorporate genome rearrangement scores might too unfavorably skew the tradeoff between quality and computation time. In any case, after further algorithmic improvements, a comparison of the ortholog quality with positional approaches such as MSOAR [8] and Shao and Moret's ILP formulations [7] is of interest.

One further possible application could be to use OAD or OCD as post-processing of ortholog clusters that are computed by other methods such as InParanoid but that are not fully resolved.

Finally, it would also be interesting to use the duplication scenario in other applications or to consider other ways of obtaining the similarity data encoded in the bipartite graph. Instead of taking only sequence information into account when computing the bipartite graph G, one might also rely on protein interaction data to compute the edge weights [10]. Using this data, our approach could also serve as a step in pairwise network alignment between two species where one has been subject to a whole-genome duplication after the speciation event.

Acknowledgments

We thank Chunfang Zheng (University of Ottawa) for helpful discussions on the data acquisition.

References

[1] D. M. Kristensen, Y. I. Wolf, A. R. Mushegian, E. V. Koonin, Computational methods for gene orthology inference, Briefings in Bioinformatics 12 (5) (2011) 379–391.

[2] E. L. L. Sonnhammer, T. Gabaldón, A. W. S. da Silva, M. J. Martin, M. Robinson-Rechavi, B. Boeckmann, P. D. Thomas, C. Dessimoz, et al., Big data and other challenges in the quest for orthologs, Bioinformatics 30 (21) (2014) 2993–2998. doi:10.1093/bioinformatics/btu492.

[3] A. M. Altenhoff, B. Boeckmann, S. Capella-Gutierrez, D. A. Dalquen, T. DeLuca, K. Forslund, J. Huerta-Cepas, B. Linard, C. Pereira, L. P. Pryszcz, F. Schreiber, A. S. Da Silva, D. Szklarczyk, C.-M. Train, P. Bork, O. Lecompte, C. v. Mering, I. Xenarios, K. Sjölander, L. J. Jensen, M. J. Martin, M. Muffato, T. Gabaldon, S. E. Lewis, P. D. Thomas, E. Sonnhammer, C. Dessimoz, Q. for Orthologs Consortium, Standardized benchmarking in the quest for orthologs, Nature methods 13 (5) (2016) 425 – 430. doi:10.1038/nmeth.3830.

[4] E. L. L. Sonnhammer, G. Östlund, InParanoid 8: Orthology analysis between 273 proteomes, mostly eukaryotic, Nucleic Acids Res. 43 (D1) (2015) D234–D239.

[5] C. Zheng, K. M. Swenson, E. Lyons, D. Sankoff, OMG! Orthologs in multiple genomes — competing graph-theoretical formulations, in: Proc. 11th WABI, Vol. 6833 of LNCS, Springer, 2011, pp. 364–375.

[6] G. Shi, L. Zhang, T. Jiang, MSOAR 2.0: Incorporating tandem duplications into ortholog assignment based on genome rearrangement, BMC Bioinformatics 11 (1) (2010) 10.

[7] M. Shao, B. M. E. Moret, Comparing genomes with rearrangements and segmental duplications, Bioinformatics 31 (12) (2015) 329–338. `doi:10.1093/bioinformatics/btv229`.

[8] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, T. Jiang, Assignment of orthologous genes via genome rearrangement, IEEE/ACM T. Comput. Bi. 2 (4) (2005) 302–315.

[9] R. L. Tatusov, N. D. Fedorova, J. D. Jackson, A. R. Jacobs, B. Kiryutin, E. V. Koonin, D. M. Krylov, R. Mazumder, S. L. Mekhedov, A. N. Nikolskaya, et al., The COG database: an updated version includes eukaryotes, BMC Bioinformatics 4 (1) (2003) 41.

[10] R. Singh, J. Xu, B. Berger, Global alignment of multiple protein interaction networks with application to functional orthology detection, Proc. Natl. Acad. Sci. 105 (35) (2008) 12763–12768.

[11] H. N. Gabow, R. E. Tarjan, Faster scaling algorithms for network problems, SIAM J. Comput. 18 (5) (1989) 1013–1036.

[12] Y. Shiloach, Another look at the degree constrained subgraph problem, Inf. Process. Lett. 12 (2) (1981) 89–92.

[13] S. Bruckner, F. Hüffner, C. Komusiewicz, R. Niedermeier, S. Thiel, J. Uhlmann, Partitioning into colorful components by minimum edge deletions, in: Proc. 23rd CPM, Vol. 7354 of LNCS, Springer, 2012, pp. 56–69.

[14] A. Adamaszek, G. Blin, A. Popa, Approximation and hardness results for the maximum edges in transitive closure problem, in: Proc. 25th IWOCA, Vol. 8986 of LNCS, Springer, 2015, pp. 13–23.

[15] A. Adamaszek, A. Popa, Algorithmic and hardness results for the colorful components problems, in: Proc. 11th LATIN, Vol. 8392 of LNCS, Springer, 2014, pp. 683–694.

[16] R. Dondi, F. Sikora, Parameterized Complexity and Approximation Issues for the Colorful Components Problems, in: Proceedings of the 12th Conference on Computability in Europe (CiE '16), Vol. 9709 of LNCS, Springer, 2016, pp. 261–270.

[17] M. Remm, C. E. V. Storm, E. L. L. Sonnhammer, Automatic clustering of orthologs and in-paralogs from pairwise species comparisons1, Journal of Molecular Biology 314 (5) (2001) 1041–1052.

[18] G. Östlund, T. Schmitt, K. Forslund, T. Köstler, D. N. Messina, S. Roopra, O. Frings, E. L. L. Sonnhammer, InParanoid 7: New algorithms and tools for eukaryotic orthology analysis, Nucleic Acids Res. 38 (suppl_1) (2010) D196–D203.

[19] A.-C. Berglund, E. Sjölund, G. Östlund, E. L. L. Sonnhammer, InParanoid 6: Eukaryotic ortholog clusters with inparalogs, Nucleic Acids Res. 36 (suppl_1) (2008) D263–D266.

[20] K. P. O'Brien, M. Remm, E. L. L. Sonnhammer, Inparanoid: A comprehensive database of eukaryotic orthologs, Nucleic Acids Res. 33 (suppl_1) (2005) D476–D480.

[21] M. Lechner, M. Hernandez-Rosales, D. Doerr, N. Wieseke, A. Thévenin, J. Stoye, R. K. Hartmann, S. J. Prohaska, P. F. Stadler, Orthology detection combining clustering and synteny for very large datasets, PLoS ONE 9 (8) (2014) 1–10.

[22] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, Saket Saurabh, Parameterized Algorithms, Springer, 2015.

[23] S. Micali, V. V. Vazirani, An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs, in: Proc. 21st FOCS, IEEE Computer Society, 1980, pp. 17–27.

[24] C. Komusiewicz, J. Uhlmann, Cluster editing with locally bounded modifications, Discrete Appl. Math. 160 (15) (2012) 2259–2270.

[25] S. Böcker, P. Damaschke, Even faster parameterized cluster deletion and cluster editing, Inf. Process. Lett. 111 (14) (2011) 717–721.

[26] J. Guo, Problem kernels for NP-complete edge deletion problems: Split and related graphs, in: Proc. 18th ISAAC, Vol. 4835 of LNCS, Springer, 2007, pp. 915–926.

[27] A. P. Chan, et al., Draft genome sequence of the oilseed species ricinus communis, Nat. biotechnol. 28 (9) (2010) 951–956.

[28] G. A. Tuskan, et al., The genome of black cottonwood, Populus trichocarpa (torr. & gray), Science 313 (5793) (2006) 1596–1604.

[29] X. Argout, et al., The genome of Theobroma cacao, Nat. genet. 43 (2) (2011) 101–108.

[30] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman, Basic local alignment search tool, J. Mol. Bio. 215 (3) (1990) 403–410.

[31] A. M. Altenhoff, B. Boeckmann, S. Capella-Gutierrez, D. A. Dalquen, T. DeLuca, K. Forslund, J. Huerta-Cepas, B. Linard, C. Pereira, L. P. Pryszcz, F. Schreiber, A. Sousa da Silva, D. Szklarczyk, C.-M. Train, P. Bork, O. Lecompte, C. von Mering, I. Xenarios, K. Sjölander, L. Juhl Jensen, M. J. Martin, M. Muffato, T. Gabaldón, S. E. Lewis, P. D. Thomas, E. Sonnhammer, C. Dessimoz, Standardised Benchmarking in the Quest for Orthologs, Nature Methods 13 (5) (2016) 425–430.

6. Appendix: Additional Tables

In this section, we give the comparison of OAD and OCD for only those connected components that were solved by both methods.

Table 6: Average sequence similarity for the predicted paralogs and percentage of clusters by OAD for which the gene tree matches the sequence tree restricted to components solved by both the OAD and OCD methods. Here d denotes the duplication bonus and $e \mid x$ the BLAST E-value at the xth percentile, % cgt denotes the percentage of size-three clusters with correct gene trees, % dc denotes the percentage size-three clusters whose predicted paralogs are located on different chromosomes of *Populus*.

d	$e \mid 75$	$e \mid 50$	$e \mid 25$	% cgt	% dc
		Ricinus vs. *Populus*, $t = 10^{-80}$			
−1.1	—	—	—	—	—
−0.5	2.06e-124	1.68e-177	0.00e+00	69.47	87.68
0	1.52e-124	5.97e-177	0.00e+00	69.02	87.66
1.0	1.61e-123	2.03e-176	0.00e+00	68.27	87.61
		Ricinus vs. *Populus*, $t = 10^{-140}$			
−1.1	—	—	—	—	—
−0.5	0.00e+00	0.00e+00	0.00e+00	84.01	89.01
0	0.00e+00	0.00e+00	0.00e+00	82.37	88.76
1.0	0.00e+00	0.00e+00	0.00e+00	82.00	88.43
		Ricinus vs. *Populus*, $t = 0$			
−1.1	—	—	—	—	—
−0.5	0.00e+00	0.00e+00	0.00e+00	95.22	88.69
0	0.00e+00	0.00e+00	0.00e+00	95.05	88.42
1.0	0.00e+00	0.00e+00	0.00e+00	94.59	88.08
		Cacao vs. *Populus*, $t = 10^{-80}$			
−1.1	—	—	—	—	—
−0.5	2.83e-117	1.93e-172	0.00e+00	67.68	88.29
0	2.31e-117	6.33e-171	0.00e+00	66.35	87.69
1.0	7.47e-117	1.91e-169	0.00e+00	65.88	87.85
		Cacao vs. *Populus*, $t = 10^{-140}$			
−1.1	—	—	—	—	—
−0.5	0.00e+00	0.00e+00	0.00e+00	83.49	88.84
0	0.00e+00	0.00e+00	0.00e+00	82.42	87.76
1.0	0.00e+00	0.00e+00	0.00e+00	81.21	87.57
		Cacao vs. *Populus*, $t = 0$			
−1.1	—	—	—	—	—
−0.5	0.00e+00	0.00e+00	0.00e+00	94.54	89.16
0	0.00e+00	0.00e+00	0.00e+00	94.63	88.64
1.0	0.00e+00	0.00e+00	0.00e+00	94.05	88.11

Table 7: Average sequence similarity for the predicted paralogs and percentage of clusters by OCD for which the gene tree matches the sequence tree restricted to components solved by both the OAD and OCD methods. Here d denotes the duplication bonus and $e \mid x$ the BLAST E-value at the xth percentile, % cgt denotes the percentage of size-three clusters with correct gene trees, % dc denotes the percentage size-three clusters whose predicted paralogs are located on different chromosomes of *Populus*.

d	$e \mid 75$	$e \mid 50$	$e \mid 25$	% cgt	% dc
Ricinus vs. *Populus*, $t = 10^{-80}$					
-1.1	3.94e-134	0.00e+00	0.00e+00	75.17	90.28
-0.5	1.16e-129	0.00e+00	0.00e+00	71.87	88.56
0	3.07e-129	0.00e+00	0.00e+00	71.61	88.50
1.0	3.52e-129	0.00e+00	0.00e+00	71.51	88.51
Ricinus vs. *Populus*, $t = 10^{-140}$					
-1.1	0.00e+00	0.00e+00	0.00e+00	88.80	89.99
-0.5	0.00e+00	0.00e+00	0.00e+00	85.65	88.68
0	0.00e+00	0.00e+00	0.00e+00	85.22	88.72
1.0	0.00e+00	0.00e+00	0.00e+00	85.02	88.66
Ricinus vs. *Populus*, $t = 0$					
-1.1	0.00e+00	0.00e+00	0.00e+00	100.00	89.70
-0.5	0.00e+00	0.00e+00	0.00e+00	98.53	88.35
0	0.00e+00	0.00e+00	0.00e+00	98.07	88.29
1.0	0.00e+00	0.00e+00	0.00e+00	97.71	88.24
Cacao vs. *Populus*, $t = 10^{-80}$					
-1.1	2.60e-134	0.00e+00	0.00e+00	75.01	90.40
-0.5	8.48e-128	5.87e-180	0.00e+00	70.96	88.56
0	6.78e-126	1.98e-178	0.00e+00	70.11	88.44
1.0	2.34e-125	3.90e-178	0.00e+00	69.88	88.33
Cacao vs. *Populus*, $t = 10^{-140}$					
-1.1	0.00e+00	0.00e+00	0.00e+00	88.60	89.26
-0.5	0.00e+00	0.00e+00	0.00e+00	85.71	87.80
0	0.00e+00	0.00e+00	0.00e+00	85.23	87.81
1.0	0.00e+00	0.00e+00	0.00e+00	84.91	87.86
Cacao vs. *Populus*, $t = 0$					
-1.1	0.00e+00	0.00e+00	0.00e+00	100.00	90.14
-0.5	0.00e+00	0.00e+00	0.00e+00	98.36	88.44
0	0.00e+00	0.00e+00	0.00e+00	97.70	88.44
1.0	0.00e+00	0.00e+00	0.00e+00	97.31	88.40