Chapter 6

POInT: A Tool for Modeling Ancient Polyploidies Using Multiple Polyploid Genomes

Yue Hao and Gavin C. Conant

Abstract

Ancient polyploidy events are widely distributed across the evolutionary history of eukaryotes. Here, we describe a likelihood-based tool, POInT (the *P*olyploidy *O*rthology *In*ference *T*ool), for modeling ancient whole genome duplications and triplications, assigning homoeologous genes to subgenomes and inferring gene losses across different parental subgenomes after polyploidy.

Key words Ancient polyploidy, Subgenomes, Gene loss, Evolution, Orthology

1 Introduction

About 14 years ago, we described a model-based approach to understanding the resolution of polyploidy events through duplicate gene loss [1, 2]. This tool, POInT (the *P*olyploidy *O*rthology *In*ference *T*ool), uses synteny data to statistically associate adjacent genes in each genome, allowing for the combination of loss information from multiple genes and genomes to "phase" regions of each polyploid genome relative to the others, identifying orthologous and paralogous chromosome regions. This phasing is performed probabilistically using a hidden Markov model (HMM; *see* [3]) that resembles the Lander-Green approach for constructing linkage maps from ordered genetic markers and a pedigree [4]. If we define each of the homoeologous genes created by the polyploidy as a "pillar" (*see* Fig. 1), we can see that, at each such pillar, POInT calculates the probability of the observed gene presence–absence data conditional upon each of the 2^n possible orthology relationships and also conditional upon a phylogenetic tree and a model for gene loss (The approach is conceptually identical for hexaploidies and octoploidies, but there are more possible orthology relationships to be tested). The HMM transition probability θ_j corresponds to the probability that orthology changes between

Alejandro Pereira-Santana, Samuel David Gamboa-Tuz and Luis Carlos Rodríguez-Zapata (eds.), *Plant Comparative Genomics*, Methods in Molecular Biology, vol. 2512, https://doi.org/10.1007/978-1-0716-2429-6_6,

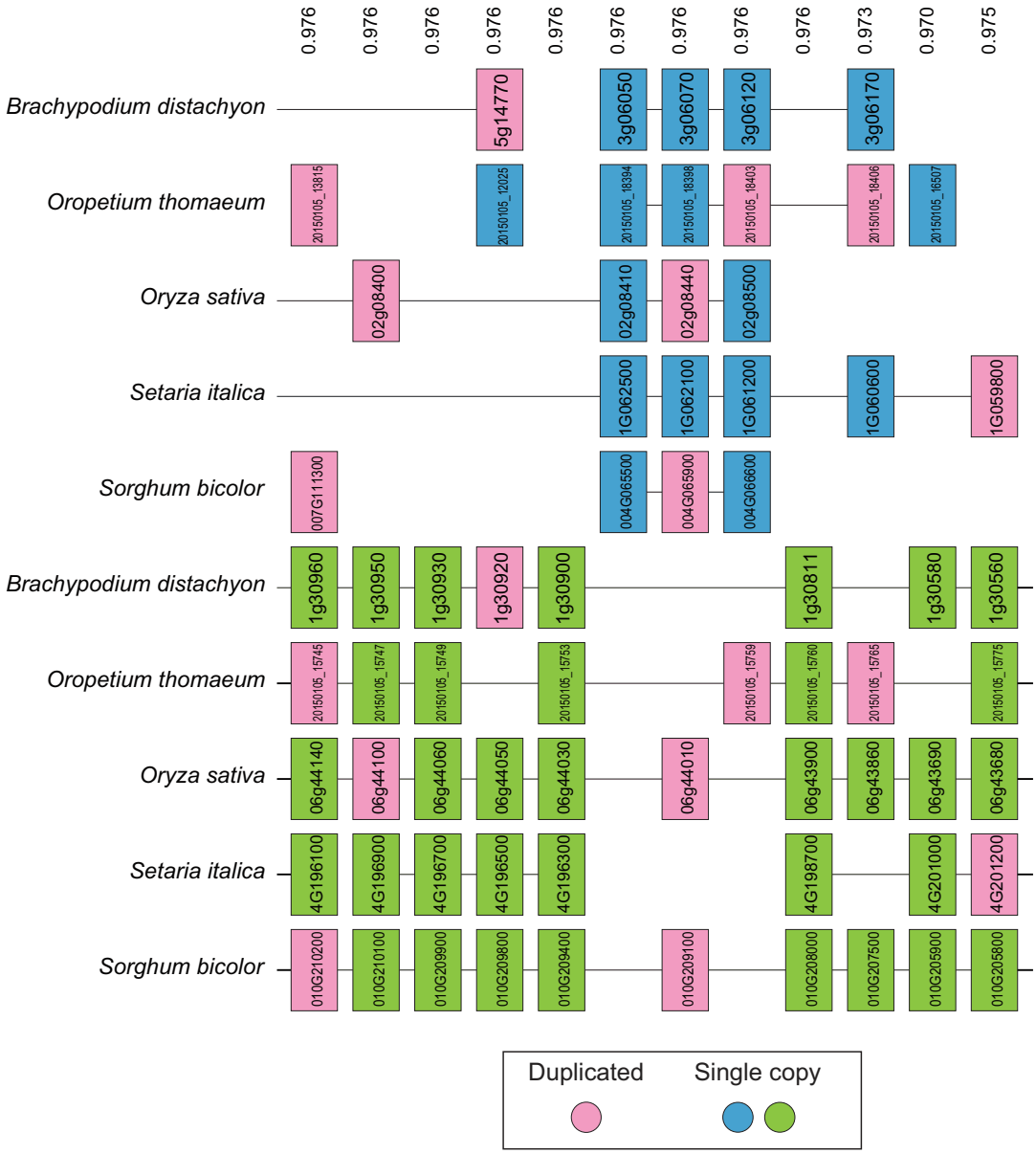

Fig. 1 Double conserved synteny blocks shared by *Brachypodium distachyon*, *Oropetium thomaeum*, *Oryza sativa*, *Setaria italica*, and *Sorghum bicolor* after the grass ρ whole genome duplication [5]. Pink genes are retained duplicates after the WGD, blue and green genes returned to single-copy and are from two different parental subgenomes. Gene synteny is indicated by the horizontal lines, and a gap represents gene loss after polyploidy. Posterior probabilities are shown on top of each pillar

syntenic neighbors at pillars $j-1$ and j, with the θ parameter is best thought of as an "error" term accounting for situations where the orthology assignments at the beginning of a synteny block differ from those at the end. This framework is how we are able to allow the presence–absence data to inform orthology relationships at

neighboring pillars. Pillars that are separated by synteny breaks are independent in their orthology relationships (i.e., $\theta_j = 1/2$). The model parameters and phylogenetic branch lengths are then fit to the pillar data using maximum likelihood and standard numerical optimization [6].

The power of this modeling framework is considerable. It produces probabilistic estimates of the orthology relationships between all of the homoeologous genes in the genome analyzed. These estimates can be used for problems such as identifying valid phylogenetic markers from polyploid genomes. We ourselves have used them as sequence-independent markers of locus history for the detection and analysis of gene conversions [7–9], as well as to link gene loss and preservation patterns after polyploidy to molecular functions [5, 10, 11] and to explore the role a genome ploidy increase played in the formation of a clade of parasitic nematodes [12]. POInT can also be used as a simulation engine: we have used it to confirm that polyploidies are indeed shared events between a number of genomes [2], test hypotheses about biases in gene loss [11] and to infer the *type* of polyploidy event present in certain genomes [12].

Our original POInT analyses were predicated on sets of homoeologous genes and an inferred ancestral genome order that the Wolfe lab developed for the yeast WGD as part of the Yeast Gene Order Browser (YGOB, http://ygob.ucd.ie) project [13, 14]. Recently, we have expanded POInT and developed a new software pipeline that allows us to examine arbitrary polyploidy events, so long as we have at least two genome sequences from species sharing the event as well as an outgroup genome lacking it [5]. This pipeline operates following a clear analogy to the process of creating datasets for phylogenetic analyses. In particular, we have an "alignment" step followed by the phylogenetic modeling step just sketched. We will generally describe running this pipeline under the assumption of a genome *duplication* (tetraploidy) for simplicity, using terms such as "duplicated regions." However, our approach is fully general, and we have also successfully applied it to hexaploidies and octoploidies.

The full suite of POInT tools is freely available from either our website (http://conantlab.org/POInT/POInT.html) or GitHub (https://github.com/gconant0/POInT). The majority of POInT is written in standard C++ (with the exception of one perl script) and has modest dependencies: the lapack linear algebra libraries for the likelihood computation [15], the GNU plotutils package (optional and used for producing visualizations), a random number generator (https://people.sc.fsu.edu/~jburkardt/f77_src/ranlib/ranlib.html), the OpenMP shared memory parallel library [16] and the Bioperl package, which is used only for inferring the initial set of gene orders in the various genomes [17]. The open-source code for the random number generator and the required lapack subroutines

are included in the distribution for convenience: on systems with lapack and blas preinstalled, the installed versions are used in preference to the copies in the distribution.

2 POInT Dataset Assembly/Synteny Block Inference

This "alignment" step seeks to assemble blocks of N-fold conserved synteny (NCS) produced by the various types of polyploidy (e.g., $N = 2$ for a tetraploidy, $N = 3$ for a hexaploidy and $N = 4$ for octoploidy). These blocks represent the products of the polyploidy and contain both surviving duplicated loci as well as regions where one or more of the homoeologs ("duplicates" from polyploidy) have been lost (Fig. 1). The inference process has three substeps: (1) homology inference, (2) inference of NCS blocks between a single polyploid genome and the nonpolyploid outgroup, and (3) the merging of NCS blocks from multiple polyploid genomes and the inference of an ancestral block order. The data required for **step 1** are as follows.

1. FASTA files with the coding regions and translations of all protein-coding genes in each polyploid genome and the nonpolyploid outgroup.
2. GFF files describing the relative contig or chromosome position of the coding regions/genes in those FASTA files for all of the genomes in question.

2.1 Step 1: Homology Inference

Our pipeline requires an outgroup or reference genome that is a relatively close relative of the polyploid genomes but which lacks the polyploidy. It is conceptually convenient to think of it as the "ancestral" prepolyploid genome, although that is not technically accurate. This homology search could be conducted in several ways, with a simple BLAST search [18] being perhaps the most obvious. Indeed, our first analysis [5] used a BLAST-like approach with low sensitivity but high computational efficiency that we developed from the SeqAn library [19]. More recently, we have found that GenomeHistory [20], a tool we developed about 20 years ago, while slow, gives good coverage of the genomes, including pairwise estimates of synonymous and nonsynonymous divergence (K_s and K_a), which are used in the next steps of the analysis. We note that the homology search only compares each polyploid genome with the nonpolyploid outgroup and with itself: we do not directly compare the polyploid genomes. When the homology search is complete, we store several pieces of information: (a) any pair of genes from the polyploid genome and the outgroup that pass homolog cutoffs in terms of percent amino acid identity of the pairwise alignment [21] of their two sequences, (b) pairs of genes both either from the polyploid or the outgroup genome that pass

similar filters and are hence potential tandem duplicates, and (c) The relative position of each of the genes in either "a" or "b" in their respective genomes. These data take the form of an ordered list of genes on contigs or chromosomes, omitting any genes with no homologs in "a" or "b."

2.2 Step 2: NCS "Scaffolding"

The next pipeline step uses the putatively single-copy genes from the nonpolyploid genome to infer the set of duplicated (or more) regions created by the polyploidy in each polyploid genome. Were all of the duplicate genes created by the polyploidy still extant, the identification of such regions would be trivial. In the face of duplicate loss, it becomes more complex.

We frame this problem as first defining a set A of n NCS blocks, each pillar A_i of which consists of one gene from the nonpolyploid outgroup ($A_i \in A | 1 \leq i \leq n$). Each A_i has elements $A_i(p_1) \ldots A_i(p_k)$, representing the k (= 2 for a tetraploidy) homologous genes created by the polyploidy. Associated to A_i are also all of the genes in the polyploid genome homologous to the nonpolyploid genome gene for that pillar $\{h_1 \ldots h_h\}$. At most k of these homologs can be assigned to $A_i(p_1) \ldots A_i(p_k)$. Finally, we define $O(A_1 \ldots A_n)$ to be the order of the pillars used for our analysis. Hence, $A_{O(i)}$ represents the ith pillar in this ordering. For a given $A_{O(i)}(p_l) | 1 \leq l \leq k$, define $A_{O(i+j)}(p_l)$ such that $j = \min(x; i + 1 \leq x \leq n)$ where $A_{O(i+x)}(p_l) \neq \emptyset$. In other words, $i + j$ is the next pillar after i in $O(A_1 \ldots A_n)$ with an assigned gene for parental subgenome l. From this framework, we can create a scoring function s for comparing different combinations of homolog assignments and pillar orders:

$$s = \sum_{i=1}^{n} \sum_{l=1}^{k} \begin{cases} 1 & A_{O(i)}(p_l) \text{ and } A_{O(i+j)}(p_l) \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

This equation indicates that s is the sum of the number of positions in $O(A_1 \ldots A_n)$ where the genes in each pillar are the genomic neighbors of the genes in the next nonempty position. One might wonder why the pillar order is estimated rather than simply being taken from the outgroup. However, in many cases, the outgroups used are rather distant relatives of the true polyploid progenitors and using these genome orders, even were the sequences in question perfectly assembled, would introduce not only all the postpolyploidy rearrangements seen in the polyploid genomes but also all those that occurred in the outgroup.

Equation 1 only allows us to score a particular combination of homeolog assignment and order (e.g., one point in a large state space). Unfortunately, the number of possible such points is enormous: there are $n!$ orders alone, without including the homeolog assignments. We therefore use simulated annealing [22, 23] to search for optimal values of s. Simulated annealing is a common

optimization approach that proposes small random changes to a current point in the state space: after each such move s is recomputed. The move is then accepted if either it improves the score or if the decrease in score is below a threshold that is tuned to decrease over the annealing run. This part of the analysis requires a certain degree of "art," as it is generally necessary to make increasingly long runs of the annealing algorithm until longer runs no longer produce meaningfully higher values of s.

From a practical perspective, this step is performed by the tool POInT_genome_scaffold in the POInT distribution. This program first collapses tandem duplicates in the outgroup and polyploid genomes (group "b" above). It also allows for a K_s or K_a filter on the homologs, allowing the user to remove distant homologs. It is similarly possible to take only the top m homologs (in terms of smallest K_s or K_a) for the analysis, a feature that is useful in genomes with high degrees of nested polyploidy. The output is a POInT-specific format that identifies the outgroup gene, the (up to) k homeologs and whether or not they have synteny support and in which direction.

2.3 Step 3: Pillar Merging and Global Order Inference

In the POInT distribution, we provide a script (POInT_merge.pl) that combines the optimal runs of POInT_genome_scaffold from across multiple genomes into a common set of pillars with at least one surviving homeolog across all of these genomes. This requirement for the presence of at least one gene in every genome limits the size of resulting datasets but is necessary because POInT cannot yet model the complete loss of an ancestral gene. The merge program allows the user to include pillars with different levels of synteny support, but we invariably only include pillars where every gene present is in synteny with at least one other (the default behavior). The genes from the outgroup genome are used as indices to allow the combination of genes from the different polyploid genomes.

POInT_merge.pl produces a set of pillars order by their genome position in the outgroup genome, since the $O(A_1 \ldots A_n)$ are different for each polyploid genome. This order will generally poor; it might, for example, include a large number of synteny breaks. Hence, the final step of the assembly pipeline is to run POInT_ances_order, which again uses simulated annealing to infer a pillar order with as few synteny breaks as possible. It does this by proposing different pillar orders and counting the resulting synteny breaks. In this case, the raw number of synteny breaks itself is used as the optimality function, and we find that a number of short annealing runs with POInT_ances_order produces orders with acceptable "tracking."

We provide a number of options with the POInT_ances_order program to provide adequate performance for polyploidies of different ages. In general, older polyploidies show more

rearrangements, meaning that the ancestral order inference is more difficult. Hence, we offer a "preoptimization" step that uses a greedy algorithm to initially order the data: this subroutine works best when the initial order is very poor (has many breaks). Second, we allow the user to control the quality of the "fixed blocks" in the inferences. Obviously, if two pillars are completely connected (show no synteny breaks between them), it is unproductive to rearrange within this "block," since the number of synteny breaks cannot improve. In fact, we know that these pillars must appear in these positions in the optimal order. Hence, POInT_ances_order performs rearrangements on these inferred blocks rather than the individual pillars: especially for datasets with few synteny breaks, this approach dramatically decreases the search space size.

When the dataset has a larger number of synteny breaks, this block approach will not reduce the search space as much: in those cases, we allow the user to specify a block cutoff *m*, corresponding to a number of synteny breaks tolerated within a block. For instance, by using a setting of -m:3 in POInT_ances_order, pairs (or more) of pillars with 3 or fewer synteny breaks will be treated as a single block and no rearrangements attempted within them. By starting with relatively larger values of this parameter, one can perform first coarse and then fine optimization (smaller *m*) and significantly reduce the search time for a reasonable ancestral order.

3 Modeling Polyploid Genome Evolution with POInT

Once these steps are complete, modeling of polyploidy events is possible. POInT itself fits Markov models of postpolyploidy gene loss to the pillars inferred above [2, 10]. In the current version, these models can be specified by the user in file format illustrated with the example in Fig. 2. Transition probabilities for these models are first computed by exponentiating the instantaneous rate matrix they define [24], and they are then fit to the pillar data using numerical optimization [6]. The models have states corresponding to single-copy, duplicated, triplicated, and quadruplicated genes, as well as the potential for variations of these states, for example, *fixed* duplicates that will remain in the duplicated state permanently. Figure 2 gives two example models we have developed for POInT: more models can be downloaded from our website (http://conantlab.org/software.html).

The running time of a POInT optimization can be significant: the algorithmic complexity of the algorithm is $O(2^{2n})$ for a tetraploidy (where n is the number of genomes) and greater for higher-level polyploidies. We have therefore implemented POInT as a parallel program using the OpenMP shared memory paradigm [16]. An analysis of ~4100 pillars across 11 taxa sharing a tetraploidy hence takes on the order of a few weeks on an Intel Phi

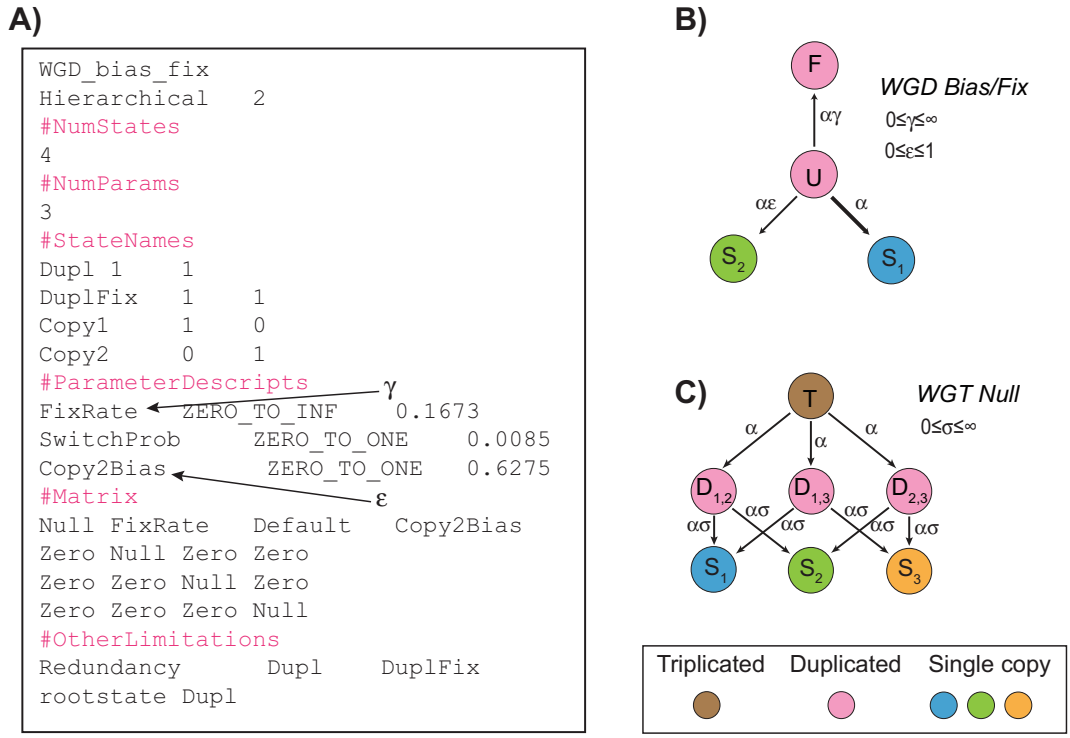

Fig. 2 Example model files required by POInT. (**a**) Description of a model file and estimated model parameters. (**b**) Example model of a WGD, *U* stands for an undifferentiated duplicated state, *F* is a fixed duplicate, and S_1 and S_2 represent single-copy states for the two parental subgenomes respectively. Here γ is the duplication retention parameter and ε is the biased fractionation parameter, while $\alpha\gamma$, $\alpha\varepsilon$, and α are transition rates between states. (**c**) Example model for a WGT. This model shows seven possible states (triplicated, duplicated, or single-copy states) after whole genome triplication and the transition rates between these states

coprocessor [25] and a roughly equivalent amount of time on 16 cores of modern high-end Intel Pentium processors.

There are a number of different ways to use POInT, and so we provide a few examples as illustrations.

Example 1 Testing the presence of duplicate fixation In the simplest use of POInT, we have a set of loci and a known phylogeny and wish to test the fit of different models of homeolog loss to these data. In this case, we would simply run POInT twice with a known input tree (provided with the -t:<Nexus treefile> option) and two different models, for instance a null model without duplicate fixation and an alternative with it. At the end of each run, a new tree file *inputtreefile.out* is created with the model parameters and the resulting log-likelihood. Since duplicate fixation is controlled by a single parameter (γ in Fig. 2), twice the difference in log-likelihood between the two models is distributed chi-square with 1 degree of freedom [26], allowing us to infer the significance (or lack thereof) of the improvement in fit from adding this parameter.

Example 2 Inferring the phylogenetic relationships between polyploid taxa If no tree is provided to POInT, it will compute the likelihood of all possible topologies for the taxa provided. Obviously, this computation could be very slow for large numbers of taxa: in our experience, it is only practical for datasets of five or fewer taxa in the context of a WGD, and three or fewer for a WGT. The topology with the highest log-likelihood is saved as a new tree file.

Example 3 Extracting orthologous genes from polyploid genomes POInT probabilistically computes all possible sets of orthology relationships between all of the loci in all of the polyploid genomes (whether or not a gene is actually present at that position in that genome). As such, an inference of the optimal orthology relationship can be made by computing the likelihood of a particular assignment relative to all of the possible assignments. Using the -p:<filename> option in POInT will result in the program saving the probability of every possible orthology relationship for every locus to that file. The header line gives the identity of all of these assignments, which can then be parsed manually or by other software.

Example 4 Testing the hypothesis of shared verses independent polyploidies When studying the yeast WGD, the question arose if the polyploidy observed in the genome of *V. polysporus* was the same event as that seen in *S. cerevisiae* [2]. In the POInT models, independent polyploidies are equivalent to having the shared root branch of the various taxa set to zero length (*V. polysporus* is the most distant relative of *S. cerevisiae* in this dataset). To test this hypothesis, we fit the pillar data in POInT while forcing the shared root branch to zero length, which is done by providing a tree file with a zero length root branch and using the argument -zerolengthfixed. We then run POInT again without this constraint. We next use the optimized tree with the forced zero-length root to simulate new genome duplications using the POInT simulation engine POInT_simulate. This program produces simulated polyploidies under an assumed model and tree topology. One can then use the main POInT code to analyze these simulations both under the forced zero-length assumption and omitting this requirement. The result is a distribution of differences in ln-likelihood for the simulated datasets that can be compared to that for the real dataset to assess if the root branch for the real dataset is inferred to be significantly nonzero, implying a shared polyploidy for the genomes in question.

Example 5 Evaluating the gene loss pattern after polyploidy - POInT can be used to statistically test for biased fractionation, that is, the preferential gene retention and unbalanced gene loss across different parental subgenomes after polyploidy. To

implement this test, POInT will be run twice, first using a null model with the biased fractionation parameter $\varepsilon = 1$ (Fig. 2). In this null model, the transition rates from the duplication state to each single-copy state are the same, modeling a scenario where gene loss in different subgenomes is equally likely. Then, in the alternative model, ε is allowed to fall between 0 and 1, introducing biased fractionation. The likelihood estimations from the two models can again be compared using a likelihood ratio test.

4 Conclusions

POInT and its associated helper programs are an extensible framework for studying the evolution of polyploid genomes. The tools are flexible in the types of polyploidy they can model and allow the user to define new Markov models of gene loss as needed. The software itself is freely available without license restrictions and runs on a variety of parallel and serial computing platforms.

Acknowledgments

The authors were supported by U.S. National Science Foundation grant NSF-IOS-1339156.

References

1. Scannell DR, Frank AC, Conant GC, Byrne KP, Woolfit M, Wolfe KH (2007) Independent sorting-out of thousands of duplicated gene pairs in two yeast species descended from a whole-genome duplication. Proc Natl Acad Sci U S A 104:8397–8402
2. Conant GC, Wolfe KH (2008) Probabilistic cross-species inference of orthologous genomic regions created by whole-genome duplication in yeast. Genetics 179:1681–1692
3. Durbin R, Eddy SR, Krogh A, Mitchison G (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge
4. Lander ES, Green P (1987) Construction of multilocus genetic linkage maps in humans. Proc Natl Acad Sci U S A 84:2363–2367
5. Emery M, Willis MMS, Hao Y, Barry K, Oakgrove K, Peng Y, Schmutz J, Lyons E, Pires JC, Edger PP, Conant GC (2018) Preferential retention of genes from one parental genome after polyploidy illustrates the nature and scope of the genomic conflicts induced by hybridization. PLoS Genet 14(3): e1007267em
6. Press WH, Teukolsky SA, Vetterling WA, Flannery BP (1992) Numerical recipes in C. Cambridge University Press, New York, NY
7. Evangelisti AM, Conant GC (2010) Nonrandom survival of gene conversions among yeast ribosomal proteins duplicated through genome doubling. Genome Biol Evol 2: 826–834
8. Scienski K, Fay JC, Conant GC (2015) Patterns of gene conversion in duplicated yeast histones suggest strong selection on a coadapted macromolecular complex. Genome Biol Evol 7(12):3249–3258
9. Casola C, Conant GC, Hahn MW (2012) Very low rate of gene conversion in the yeast genome. Mol Biol Evol 29(12):3817–3826
10. Conant GC (2014) Comparative genomics as a time machine: how relative gene dosage and metabolic requirements shaped the time-dependent resolution of yeast polyploidy. Mol Biol Evol 31(12):3184–3193
11. Conant GC (2020) The lasting after-effects of an ancient polyploidy on the genomes of teleosts. PLoS One 15(4):e0231356

12. Schoonmaker A, Hao Y, Bird D, Conant GC (2020) A single, shared triploidy in three species of parasitic nematodes. G3 10:225–233
13. Byrne KP, Wolfe KH (2005) The yeast gene order browser: combining curated homology and syntenic context reveals gene fate in polyploid species. Genome Res 15(10):1456–1461
14. Gordon JL, Byrne KP, Wolfe KH (2009) Additions, losses and rearrangements on the evolutionary route from a reconstructed ancestor to the modern *Saccharomyces cerevisiae* genome. PLoS Genet 5(5):e1000485
15. Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D (1999) LAPACK Users' Guide, 3rd edn. Society for Industrial and Applied Mathematics, Philadelphia, PA
16. Dagum L, Menon R (1998) OpenMP: an industry standard API for shared-memory programming. IEEE Comput Sci Eng 5(1):46–55
17. Stajich JE, Block D, Boulez K, Brenner SE, Chervitz SA, Dagdigian C, Fuellen G, Gilbert JG, Korf I, Lapp H, Lehvaslaiho H, Matsalla C, Mungall CJ, Osborne BI, Pocock MR, Schattner P, Senger M, Stein LD, Stupka E, Wilkinson MD, Birney E (2002) The Bioperl toolkit: Perl modules for the life sciences. Genome Res 12(10):1611–1618
18. Altschul SF, Madden TL, Schaffer AA, Zhang JH, Zhang Z, Miller W, Lipman DJ (1997) Gapped blast and Psi-blast: a new-generation of protein database search programs. Nucleic Acids Res 25(17):3389–3402
19. Doring A, Weese D, Rausch T, Reinert K (2008) SeqAn an efficient, generic C++ library for sequence analysis. BMC Bioinformatics 9: 11
20. Conant GC, Wagner A (2002) GenomeHistory: a software tool and its application to fully sequenced genomes. Nucleic Acids Res 30(15):3378–3386
21. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48:443–453
22. Kirkpatrick S, Gelatt CDJ, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
23. Conant GC, Wolfe KH (2006) Functional partitioning of yeast co-expression networks after genome duplication. PLoS Biol 4:e109
24. Muse SV, Gaut BS (1994) A likelihood approach for comparing synonymous and nonsynonymous nucleotide substitution rates, with application to the chloroplast genome. Mol Biol Evol 11(5):715–724
25. Jeffers J, Reinders J (2013) Intel Xeon Phi coprocessor high performance programming. Morgan Kaufmann, Waltham, MA
26. Sokal RR, Rohlf FJ (1995) Biometry, 3rd edn. W. H. Freeman and Company, New York, NY