1 SUPPLEMENT

1.1 Proof of Lemma 2.8

First, we give intuition for the recurrence in Lemma 2.8, and then we sketch a proof of its correctness. We note that the proof of correctness for Lemma 2.8 mirrors, in many ways, the proof of correctness of Theorem 1 in (Kahn *et al.*, 2010) that gives a recurrence for computing a minimum-length feasible generator for a source string X and a target string Y; here, instead, we want to count the total number of feasible generators Ψ_X that have a fixed length k.

We state a lemma that we proved in (Kahn *et al.*, 2010) that describes an important structural property of the subsequences comprising a feasible generator Ψ_X.

Lemma 1.1 (Non-overlapping Property, (Kahn *et al.*, 2010)). *Consider a source string X and a sequence of duplicate operations of the form $\delta_X(s_i, t_i, p_i)$ that generates the final target string Y from an initially empty target string. The substrings X_{s_i,t_i} of X that are duplicated during the construction of Y appear as mutually non-overlapping (Def. 2.3) subsequences of Y.*

The recurrence for computing $N_X^{(k)}(Y)$ is efficient because the non-overlapping property allows us to subdivide the characters of the target string Y into independent subproblems. For example, if we are considering the set of feasible generators that contain some subsequence S of Y, $\mathcal{S}_S = \{\Psi_X : S \in \Psi_X\}$, then for every $\Psi_X \in \mathcal{S}_S$, all other elements of Ψ_X cannot overlap the characters in S. Therefore, the substrings of Y in between successive characters of S define subproblems that can be computed independently.

Note that every character of Y must appear at least once in X. In order to count the number of feasible generators Ψ_X with length $k > 1$, we must consider all subsequences of Y that could have been generated by a single duplicate operation and the number of ways we could combine exactly k of those subsequences to form a feasible generator Ψ_X. The recurrence is based on the observation that in any feasible generator, Ψ_X, y_1 must be the first (i.e. leftmost) character in some element of Ψ_X. There are then two cases to consider: either (1) y_1 was the last (or rightmost) character in the substring that was duplicated from X to generate y_1, or (2) y_1 was not the last character in the substring that was duplicated from X to generate y_1.

Proof. (sketch)

The recurrence defines two quantities: $N_X^{(k)}(Y)$ and $N_X^{(k)}(Y, i)$. We shall show, by induction, on $|Y|$ and k that for a pair of strings, X and Y, the value $N_X^{(k)}(Y)$ is equal to the number of length-k feasible generators Ψ_X, and that $N_X^{(k)}(Y, i)$ is equal to the number of length-k feasible generators Ψ_X under the restriction that the character y_1 is copied from index i in X, i.e. x_i *generates* y_1. $N_X^{(k)}(Y)$ is computed by summing over all characters x_i of X that can generate y_1..

As described above, we must consider two possibilities in order to compute $N_X^{(k)}(Y)$. In every feasible generator Ψ_X, the character y_1 must appear in some subsequence $S_{y_1} \in \Psi_X$ of Y that contains y_1 as a leftmost character and that corresponds to a substring of X that was copied conjointly to produce the subsequence S_{y_1}. Either:

- Case 1: y_1 was the last (or rightmost) character in the substring of X that was copied to produce y_1, i.e. S_{y_1} has length 1, or
- Case 2: x_{i+1} is also copied in the same duplicate operation as x_i, possibly along with other characters as well, i.e. S_{y_1} has length greater than 1.

For case one, number of length-k feasible generators Ψ_X is equal to the number of length-$(k-1)$ feasible generators $\Psi_X(Y_{2,|Y|})$ for source string X and target string $Y_{2,|Y|}$ (the suffix of Y); the union of the subsequence corresponding to the single character y_1 and any length-$(k-1)$ feasible generator $\Psi_X(Y_{2,|Y|})$ results in a length-k feasible generator Ψ_X. For case two, Lemma 1.1 implies that the total number of length-k feasible generators Ψ_X is the product of two independent subproblems. Specifically, for each $j > 1$ such that $x_{i+1} = y_j$ and for each $l \in \{1, 2, \ldots, k\}$, we compute: (i) number of length-l feasible generators for source string X and target string $Y_{2,j-1}$, namely $N_X^{(l)}(Y_{2,j-1})$, and (ii) the number of length-$(k-l)$ feasible generators for source string X and target string $y_1 Y_{j,|Y|}$ that include an element S_{y_1} in which y_1 is generated by x_i. To compute the latter, recall that all relevant feasible generators (corresponding to case 2 above) Ψ_X must contain an element that corresponds to a duplicate operation in which x_i and x_{i+1} are copied conjointly. The number of relevant length-$(k-l)$ feasible generators for source string X and target string $y_1 Y_{j,|Y|}$ that contain an element S_{y_1} that corresponds to a substring of X starting at x_i and also containing x_{i+1} is equal to the number of relevant length-$(k-l)$ feasible generators for source string X and target string $Y_{j,|Y|}$ that contain some element S_{y_j} that corresponds to a substring of X starting at x_{i+1}, namely $N_X^{(k-1)}(Y_{j,|Y|}, i+1)$. □

1.2 The Score Function, σ

We define the score of a generator $\omega(\Psi_X)$ to be some function that reflects the biological plausibility of the event of choosing a particular generator Ψ_X from the space of all generators and then duplicating the substrings of Ψ_X in some duplication scenario. When inferring a sequence of duplicate operations that can account for the construction of a particular target string Y by copying substrings of a particular source string X, a reasonable assumption is that the "simplest" explanation is the best. We consider the most-parsimonious duplication scenario–that is, the one requiring the fewest number of

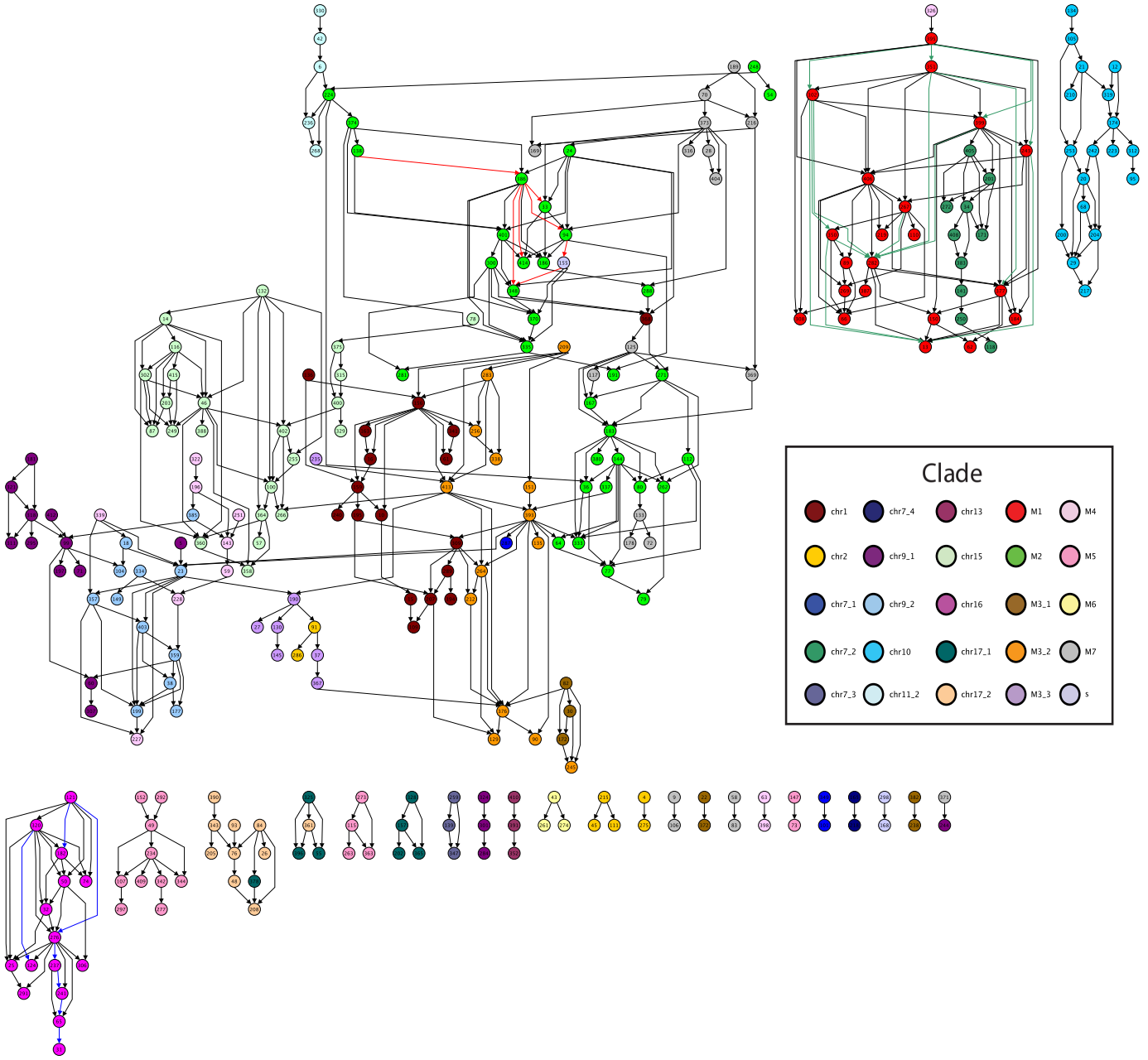

Fig. 1. The maximum parsimony DAG for a set of 391 duplication blocks in the human genome. The nodes represent duplication blocks. Edges indicate evolutionary relations; an edge is directed from a node u to a node v if the most-parsimonious duplication scenario includes duplication events that copy substrings of u in the construction of v. Jiang *et al.* (2007) partitioned the duplication blocks into a set of 24 clades (plus one 's' group of duplication blocks found in subtelomeric regions) that we indicate here with 25 colors on nodes. The 3 sets of colored edges represent inheritance networks for 3 conserved subsequences of duplicons. These inheritance networks are almost entirely confined to a single clade each. The green edges represent the inheritance of the duplicon sequence $[6968, 6967, 6965, 6963, 6962, 6960]$ in clade 'M1', the red edges represent the inheritance of $[7039, 7036, 7037]$ in clade 'M2', and the blue edges represent the inheritance of $[9448, 9449]$ in clade 'chr16.'

duplicate operations–to be the simplest. As noted above, the most-parsimonious solution can be computed using the duplication distance algorithm presented in Kahn and Raphael (2008, 2009). Therefore, our first consideration for scoring a generator is that one with small length, e.g. with length equal to the duplication distance, ought to have a good score.

Given a source string X, and two different target strings Y_1 and Y_2, where $|Y_1| < |Y_2|$, we assume that if the character contents of Y_1 and Y_2 are similar, then the construction of Y_1 from X is more likely than the construction of Y_2 from X. Again, this assumption favors simplicity. Therefore, two generators with length k that are feasible for Y_1 and Y_2, respectively, should be scored in a way that the generator for Y_1 is preferable to that for Y_2.

Theorems 2.7 and 2.9 allow for a score function of the form $\sigma(|\Psi_X|, l(\Psi_X))$. However, we impose two additional conditions that are biologically plausible.

1. For integers $k_1 < k_2$, $\sigma(k_1, l(\Psi_X)) > \sigma(k_2, l(\Psi_X))$. This property matches our intuition that a feasible generator with lesser cardinality (corresponding to a shorter sequence of duplicate operations needed to construct the target string) be more likely than a feasible generator with higher cardinality.
2. For identical source and target strings, $X = Y$, of length $|X| = |Y| > 1$, $\sigma(1, |Y|) > \sum_{\Psi_X : |\Psi_X| = k} \sigma(k, |Y|)$ for any $k > 1$. This will ensure that the event F of choosing a length-k generator for any $k > 1$ is less probable than choosing a length-1 generator; i.e. $Pr[F|Y, X, k] < Pr[F|Y, X, 1]$. This matches our intuition that the unique feasible generator of length 1 corresponding to the construction of Y by simply duplicating all of X in a single operation, will have higher probability than the combination of all feasible generators of length $k > 1$. Note that when $X \neq Y$, this property also ensures an analogous preference of feasible generators containing any long, contiguous substring $X_{s,t}$ that appears as a substring of Y over feasible generators that contain fragmented portions of $X_{s,t}$ to generate the same substring in Y, all other elements being equal.

A suitable score function that meets these criteria is:

$$\sigma(k, |Y|) = \frac{1}{|Y|^k}. \quad (1)$$

Undoubtedly, there are other biologically motivated score functions that may produce meaningful results.

1.3 Simulated Annealing Heuristic

(Giudici and Castelo, 2003) describes an elegant approach for moving locally in the space of DAGs via three types of simple moves- *adding* a new edge, *removing* an existing one, or *reversing* an existing one.

Definition 2.2 Given DAG $G = (V, E)$ we call the DAG $G' = (V, E')$ **neighboring** of G if and only if we can obtain G' from G with a single move- adding a new edge, removing, or reversing an existing edge.

Definition 2.2 Given an objective function f and two DAGs G_1, G_2 we call $\Delta G = f(G_1) - f(G_2)$ the difference in their energies.

Now, given a DAG $G = (V, E)$ and a random move proposed by the Simulated Annealing we:
1. Examine whether the move is legal (i.e. does not induce a cycle)
2. Decide wheter to accept the move based on the probability $p = exp(\frac{-\Delta G}{T})$
3. Perform the move

To decide whether to accept a move or not we need to compute $p = exp(\frac{-\Delta G}{T})$, where T is a temperature parameter. Then, we compare p with a random number in the interval $(0, 1)$ and if $p > rand(0, 1)$ we accept the move. We note that depending on the complexity of the objective function $f(G)$ computing ΔG could be very expensive. In fact, this is the case for the max likelihood reconstruction because computing $Pr[Y|X, k]$ takes in the worst-case $O(|Y|^3|X|k^2)$. Therefore, we employ a hashtable to store the cost of every move we have examined. As we do hundreds of independent trials we may often need to examine the same move multiple times, and the hashtable helps significantly speed up the search for good moves..

In our implementation we employ an exponential cooling schedule schedule. The temperature is updated via the equation $T_{t+1} = T_t\alpha$. We determined empirically that $\alpha = 0.98$ performs best in terms of efficiency and time.

The simulated annealing heuristic often terminated in local optima. For a particular instance, the solutions found by all 300 trials would include many globally suboptimal solutions. However, many of the locally optimal solutions encountered were "close" to the score for the best solution found. For example, the search for the max parsimony evolutionary history given in Fig. 3(a) resulted in a component whose objective score is 397; more than 1/6 of the total trials returned solutions whose objective scores are no more than 407 and well over 1/2 of the total trials returned solutions whose objective scores are no more than 437 (see Fig. 2).

REFERENCES

Giudici, P. and Castelo, R. (2003). Improving markov chain monte carlo model search for data mining. *Machine Learning*, **50**(1-2), 127–158.

Jiang, Z., Tang, H., Ventura, M., Cardone, M. F., Marques-Bonet, T., She, X., Pevzner, P. A., and Eichler, E. E. (2007). Ancestral reconstruction of segmental duplications reveals punctuated cores of human genome evolution. *Nature Genetics*, **39**, 1361–1368.

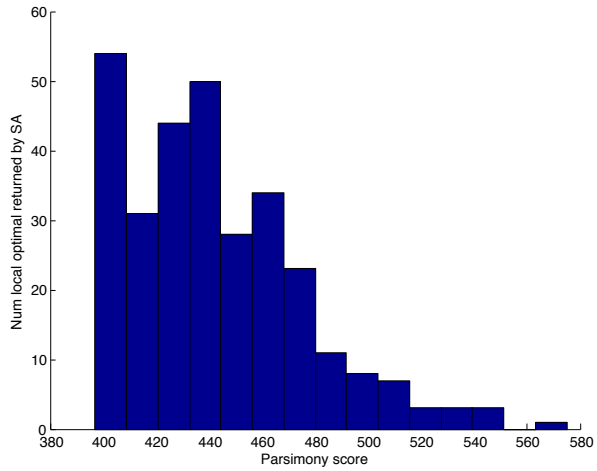

Fig. 2. Results of 300 trials of simulated annealing (SA) heuristic: number of local optima returned by SA vs. objective scores. Results are from search for max parsimony evolutionary history for component comprised of duplication blocks from clade 'chr16' whose global optimum is given in Fig.3(a).

Kahn, C., Mozes, S., and Raphael, B. (2010). Efficient algorithms for analyzing segmental duplications with deletions and inversions in genomes. *Algorithms for Molecular Biology*, **5**(1), 11.

Kahn, C. L. and Raphael, B. J. (2008). Analysis of Segmental Duplications via Duplication Distance. *Bioinformatics*, **24**, i133–138.

Kahn, C. L. and Raphael, B. J. (2009). A Parsimony Approach to Analysis of Human Segmental Duplications. In *Pacific Symposium on Biocomputing*, pages 126–137.