

Dollo-CDP: A polynomial-time algorithm for the clade-constrained large Dollo parsimony problem

SUPPLEMENTARY MATERIALS

Junyan Dai, Tobias Rubel, Yunheng Han, and Erin K. Molloy

October 16, 2023

Contents

1	Algorithms	2
2	Software Commands	2
2.1	PAUP* command for running branch-and-bound	3
2.2	PAUP* command for running the fast heuristic search (FastH)	4
2.3	PAUP* command for running the slow heuristic search (SlowH)	4
2.4	Dollo-CDP command	4
2.5	PAUP* command for computing the Dollo criterion score	4

1 Algorithms

Algorithm 1 GetState

Input: Subtree bipartition $X|Y$, the set S of n species, and an $n \times k$ character matrix $\mathbf{C}$, with $\mathbf{C}[i, j]$ indicating the state assigned to leaf i for character j

Output: State assignment for subtree bipartition $X|Y$ for each of the k characters in $\mathbf{C}$

```

1: function GETSTATE( $X|Y, S, \mathbf{C}$ )
2:    $states \leftarrow$  an array of length  $k$  containing all 0's ;  $A \leftarrow X \cup Y$ ;  $Z \leftarrow S \setminus A$ 
3:   for  $i \in \{0, 1, 2, \dots, k-1\}$  do
4:     if for all  $x \in A$ ,  $\mathbf{C}[x, i] = ?$  then  $states[i] \leftarrow ?$ 
5:     else
6:        $flag \leftarrow 0$ 
7:       if  $\exists x \in X$  s.t.  $\mathbf{C}[x, i] = 1$  then  $flag \leftarrow flag + 1$ 
8:       if  $\exists x \in Y$  s.t.  $\mathbf{C}[x, i] = 1$  then  $flag \leftarrow flag + 1$ 
9:       if  $\exists x \in Z$  s.t.  $\mathbf{C}[x, i] = 1$  then  $flag \leftarrow flag + 1$ 
10:      if  $flag \geq 2$  then  $states[i] \leftarrow 1$ 
11:   return  $states$ 

```

Algorithm 2 CountLosses

Input: Three k -vectors of state assignments: $statesU$, $statesV$, $statesW$

Output: Number of losses assuming that $statesU$ are associated with a vertex u and $statesV$ and $statesW$ are associated with children of u

```

1: function COUNTLOSSES( $statesU, statesV, statesW$ )
2:    $nlosses \leftarrow 0$ 
3:   for  $i \in \{0, 1, 2, \dots, k-1\}$  do
4:     if  $statesU[i] = 1$  and  $statesV[i] = 0$  then  $nlosses \leftarrow nlosses + 1$ 
5:     if  $statesU[i] = 1$  and  $statesW[i] = 0$  then  $nlosses \leftarrow nlosses + 1$ 
6:   return  $nlosses$ 

```

Algorithm 3 Construct subtree bipartitions from clades

Input: Set Σ of clades

Output: Subtree bipartitions allowed from Σ stored as a dictionary, where $SubBip[A]$ is a list of the allowed subtree bipartitions for clade A (note that only one side of the subtree bipartition is stored)

```

1: function CONSTRUCTSUBBIPSFROMCLADES( $\Sigma$ )
2:   Sort  $\Sigma$  by cardinality from least to greatest
3:   for  $i \in \{0, 1, \dots, |\Sigma| - 1\}$  do
4:      $A \leftarrow \Sigma[i]$ ;  $SubBip[A] \leftarrow []$ 
5:     for  $j \in \{0, 1, \dots, i-1\}$  do
6:        $X \leftarrow \Sigma[j]$ 
7:       if  $X \subset A$  then  $SubBip[A].append(X)$  (already have ptr to  $SubBip[A]$ )
8:   return  $SubBip$ 

```

2 Software Commands

In all commands for heuristic search, we specified seeds so that the results would be reproducible.

Algorithm 4 Dynamic Programming for CC-LDP

Input: Set Σ of clades (sorted by cardinality from least to greatest), a dictionary *SubBip* of allowed subtree bipartitions previously computed from Σ , an $n \times k$ character matrix **C**, with each character on species set S

Output: Fills in the dynamic programming matrix *Dollo* and the traceback matrix *TraceBack*

```
1: Lab  $\leftarrow$  dict()
2: SubBipByLab  $\leftarrow$  dict()
3: for  $A \in \Sigma$  do
4:   if  $|A| = 1$  then
5:     Do base case for leaves
6:      $st \leftarrow \mathbf{C}[A, :]$ 
7:     Dollo[ $A, st$ ]  $\leftarrow 0$ 
8:     Lab[ $A$ ]  $\leftarrow st$ 
9:     SubBipByLab[ $A$ ][ $st$ ]  $\leftarrow \emptyset$ 
10:    Traceback[ $A$ ][ $st$ ]  $\leftarrow NULL$ 
11:   else
12:     Find state assignments for clade  $A$  and their associated subtree bipartitions
13:     Lab[ $A$ ]  $\leftarrow \emptyset$ 
14:     SubBipByLab[ $A$ ]  $\leftarrow$  dict()
15:     for  $X \in \text{SubBip}[A]$  do
16:        $Y \leftarrow A \setminus X$ 
17:        $st \leftarrow \text{GetState}(X|Y, S, \mathbf{C})$ 
18:       Add  $st$  to set Lab[ $A$ ]
19:       Add  $X$  to set SubBipByLab[ $A$ ][ $st$ ] (initialize if it does not exist)
20:     For each unique state assignment fill in DP matrix
21:      $bestDollo \leftarrow \infty$ 
22:      $bestX, bestY, bestXState, bestYState \leftarrow NULL$ 
23:     for  $st \in \text{Lab}[A]$  do
24:       for  $X \in \text{SubBipByLab}[A][st]$  do
25:          $Y \leftarrow A \setminus X$ 
26:         for  $St_X \in \text{Lab}[X]$  do
27:           for  $St_Y \in \text{Lab}[Y]$  do
28:              $cX \leftarrow Dollo[X, St_X]$ 
29:              $cY \leftarrow Dollo[Y, St_Y]$ 
30:              $cA \leftarrow \text{CountLosses}(st, St_X, St_Y)$ 
31:              $score \leftarrow cX + cY + cA$ 
32:             if  $bestDollo > score$  then
33:                $bestDollo \leftarrow score$ 
34:                $bestX \leftarrow X$ 
35:                $bestXState \leftarrow St_X$ 
36:                $bestY \leftarrow Y$ 
37:                $bestYState \leftarrow St_Y$ 
38:            $Dollo[A, st] \leftarrow bestDollo$ 
39:           TraceBack( $A, st$ )  $\leftarrow (bestX, bestXState, bestY, bestYState)$  (two children)
```

2.1 PAUP* command for running branch-and-bound

```
#NEXUS
BEGIN PAUP;
set autoclose=yes warntree=no warnreset=no;
execute <character matrix nexus file>;
outgroup <outgroup name>;
ctype dollo:1-<total number of characters>;
bandb;
```

```

rootTrees;
savetrees File=<output file> root=yes trees=all format=newick;
END;

```

2.2 PAUP* command for running the fast heuristic search (FastH)

```

#NEXUS
BEGIN PAUP;
set autoclose=yes warntree=no warnreset=no;
execute <character matrix nexus file>;
outgroup <outgroup name>;
ctype dollo:1-<total number of characters>;
hsearch start=stepwise addSeq=random swap=None nreps=10 rseed=55555;
hsearch start=1 swap=TBR nbest=100 rseed=12345;
rootTrees;
savetrees File=<output file> root=yes trees=all format=newick;
END;

```

2.3 PAUP* command for running the slow heuristic search (SlowH)

```

#NEXUS
BEGIN PAUP;
set autoclose=yes warntree=no warnreset=no;
execute <character matrix nexus file>;
outgroup <outgroup name>;
ctype dollo:1-<total number of characters>;
hsearch start=stepwise addSeq=random swap=TBR nreps=100 rseed=12345;
rootTrees;
savetrees File=<output file> root=yes trees=all format=newick;
END;

```

2.4 Dollo-CDP command

```

./dollo-cdp \
-i <character matrix nexus file> \
-g <outgroup> \
-t <best 100 trees found from fast heuristic search> \
-o <output file name>

```

2.5 PAUP* command for computing the Dollo criterion score

```

#NEXUS
BEGIN PAUP;
set autoclose=yes warntree=no warnreset=no;
execute <character matrix nexus file>;
execute <tree nexus file file>;
set criterion=parsimony;
ctype dollo:1-<total number of characters>;
pscores / single=var;
END;

```