INITIAL ANALYSIS OF USING TREE HISTORIES TO SPEED UP TREE SPACE EXPLORATION

Julie Albrecht
Math and Computer Science Department
Bemidji State University
Julie.Albrecht@st.bemidjistate.edu

Reshmi Nath
Math and Computer Science Department
Bemidji State University
reshmi.nath@st.bemidjistate.edu

Abstract

TrExML is a computer program that does maximum-likelihood analysis of nucleotide sequence to infer phylogenetic relationships among species. TrExML generates the likelihood of all trees up to a given number of sequences and then adds another sequence to the list of most likely trees. Each new tree is computed and then it is added to the list of best trees containing the additional sequence. The maximum size of the best list is fixed. Thus, as each new sequence is added, many of the trees considered are thrown out and not used. We have identified that under certain conditions it is not possible to terminate computation of the new best list early without sacrificing the quality of the results.

Introduction

Maximum-likelihood analysis of nucleotide sequence data (DNA or RNA) is one of a number of techniques used to infer phylogenetic relationships among species. Maximum-likelihood programs typically begin by taking a small subset of the nucleotide sequences and arranging them in the most likely way. Then, each of the remaining sequences is added one by one until all of the sequences have been added. A final tree represents an evolutionary hypothesis and each tree generated along the way belongs to a history of the final tree. This project involves analyzing the histories of trees in a maximum-likelihood based program (called TrExML) for generating evolutionary trees in order to increase its efficiency.

Statement of problem

TrExML performs its computation by generating all trees up to a given number of sequences (given by the A switch in the datafile) and keeping a certain number in the current list of best trees (given by the K switch). Then another sequence is added in every possible way to each tree on the best list. Each new tree is evaluated in terms of its likelihood and then added to a new best list of trees containing the one additional sequence. Since maximum size of the best list is fixed, many of the trees considered are thrown out and not retained. As the K value is increased and the number of sequences increases, these computations take a long time to complete--sometimes in the order of days or weeks. Through this project we have begun to identify techniques to terminate computation of the new best list early without significantly sacrificing the quality of the results. Ideally, these techniques will offer the user of the program the opportunity to trade off the thoroughness of the tree-space exploration for the efficiency of the search.

Description of solution

As indicated earlier, each tree in the final best list has a collection of precursor trees that constitute its history. We began this project by modifying the program to track the position of each precursor tree in its best list. Furthermore, only some of the trees in the final best list are of interest. The tree at the start of the best list is the most likely. The rest of the trees are evaluated and tested to ascertain whether they are statistically significantly different than the best tree. We are only interested in those that are not. We call those trees good trees.

At this point, we have ran the program on various datafiles with different K values and have come up with some interesting observations, suggesting that for fairly large runs of the program (K>50000) at least for the last few (which is a function of the number of sequences in the input file) cycles of the computation the best list, no good trees would be eliminated. This observation suggested that it might be feasible to stop the computation at a certain point, say half way through, for the last few runs of the computation.

Description of ongoing work

We are collecting and analyzing more data in order to be certain about the cutoffs. The following is a table of sample data acquired by running the same input file with different K values. We then eliminated all trees from the best list after the last good tree. The first row of each two-row group starts with the K value. Column two in the first row is the

rank of the last good tree in the final best list. The third column gives us the highest rank this tree's precursors reached at any point through the computation cycle. The rest of the columns are the ranks that this tree's precursors reached in each computation lifecycle. For constructing the second row we found out the highest rank that each tree's precursors reached for each computation cycle. This gives us an idea of what should be the cutoff for each computation cycle. As we can see from the data only the last three columns (in this case) are worth considering for a cutoff since all the other ones are too near to K value.

K	rank of last good tree in best list	highest number rank this tree reached	Number of Sequences 8	9	10	11	12	13	14	15	16
100000	9024	99150	9726	99150	99116	99115	95881	94074	13440	8342	9024
			10394	99999	99999	99999	99992	99977	49683	43918	9024
90000	9024	89485	9809	89485	89452	89460	81902	82280	36105	31875	9024
			10394	89999	89999	89999	89994	89985	60636	70787	9024
80000	9024	79686	10090	79675	79686	79662	76257	76847	13605	18079	9024
			10394	79999	79999	79999	79999	79996	51513	62571	9024
70000	9024	69839	10769	69743	69803	69839	67692	67310	19143	17652	9024
			10394	69999	69999	69999	69999	69999	45698	55980	9024
60000	9024	59875	10173	59875	59818	59792	59017	58366	1538	9699	9024
			10394	59999	59999	59999	59980	59999	39821	46902	7404
50000	9024	49762	10308	49625	49755	49762	48569	48850	9084	9732	9024
			10394	49999	49999	49999	49988	49965	44632	43490	9024
40000	9024	39777	10319	39617	39757	39777	37386	37742	21543	21200	9024
			10394	39999	39999	39999	39995	39986	36107	36155	9024
30000	9024	29876	10216	29876	29810	29768	29162	28544	18216	17497	9024
			10394	29999	29999	29999	29999	29996	29824	29813	9024
20000	9024	19972	10391	19914	19971	19972	19047	19381	10216	8573	9024
			10394	19999	19999	19999	19999	19999	19954	19946	9024

Unfortunately, some of the data we have collected suggests that the ordering in which the sequences are considered can have a significant impact on the pattern of the output. The following is a table of data acquired from running the same file as the table before with K

= 100000. The only difference between these files is that they had an extra switch added to them in order to randomize the order in which sequences are considered. Each two-row group represents a different randomized order.

J	rank of last good tree in best list	highest number rank this tree reached	Number of Sequences 8	9	10	11	12	13	14	15	16
675675	9024	99945	10380	99945	98655	98513	98705	98767	92295	10843	9024
			10394	99999	99999	99999	99998	99998	99997	61404	9024
987845	9024	99903	10374	99903	98919	94881	95150	95007	92272	29289	9024
			10394	99999	99999	99999	99999	99998	99997	99999	9024
35895	9024	99921	10325	99921	99095	98939	97855	89080	88451	87467	9024
			10394	99999	99999	99999	99998	99991	99985	94255	9024
756891	9024	99983	10378	99983	99964	98051	95334	66702	55184	29937	9024
			10394	99999	99999	99999	99999	99999	99998	96210	9024
100000	9024	93965	4630	93665	93965	93107	93392	95301	92262	18252	9024
			10394	99999	99999	99999	99999	99999	99998	99999	9024
423789	9024	99946	10348	99946	99053	98972	90121	88941	3497	5864	9024
			10394	99999	99999	99994	99976	99992	63277	59089	9024

It is evident from the above output that as we change the J values the pattern of the output i.e. the possible cutoff point seems to change. For the last two rows (i.e. with J= 423789) the cutoff is somewhere in the 60000, while for the other J values it is way up in the 99000 range. This might seem strange at first since apparently there is nothing special about the number 432789, neither is it the smallest or largest or different from the other numbers in any other obvious way. So only logical answer seems to be that it must be the most intelligent way the taxa for this particular file could be considered, compared to the other orderings (i.e. the other J values).

Conclusion

Our initial analysis indicates that it is not possible to effectively speed up computation of the final collection of good trees by selectively stopping computation in early rounds (early cutoff). However, further analysis may yield insight into the percentage of good trees omitted from the final collection of good trees when using early cutoff. We will continue our data collection and analysis to determine these relationships. Clearly, it is important to consider reordering the input files as our data has shown that a bad ordering does not make early cutoff a good choice if we intend to collect all of the good trees. However, our reordering data has also shown that that intelligent ordering of taxa in the input file may allow the use of the early cutoff methodology. Perhaps a small initial run may suggest an ordering to use in a large run (i.e., a large K value). Another avenue for exploration involves testing statistical significance after computing each best list (i.e., finding good trees among the precursors) and using that information to help decide the early cutoff points.