Homology searches for structural RNAs: from proof of principle to practical use

SEAN R. EDDY
Janelia Research Campus, Ashburn, Virginia 20147, USA

If you search a sequence database for homologs of a structural RNA, you don't want to search just for linear sequence similarity; you also want the search program to consider whether a candidate sequence can be folded into a similar base-paired secondary structure. A powerful and general class of computational methods for combining primary sequence and secondary structure information in RNA homology searches was independently introduced just over 20 years ago by Yasu Sakakibara (working in the lab of David Haussler) and by myself (working as a postdoc with Richard Durbin). The 20th anniversary of the founding of the *RNA* journal seems a good occasion to look back at the 20 year development arc of "stochastic context-free grammar" (SCFG) methods and software for structural RNA homology searches.

My interest in RNA sequence analysis started when I was working on the three catalytic group I introns in bacteriophage T4. The T4 introns were discovered in the mid-1980's by Marlene Belfort, David Shub, and others, not long after Tom Cech's lab had made the landmark discovery that the *Tetrahymena* group I intron is a self-splicing catalytic RNA. When I arrived at the University of Colorado at Boulder as a new PhD student, it was easy to get caught up in the excitement about catalytic RNAs and group I introns, especially amongst the close-knit "RNA Club" labs at Boulder, including the Cech and Uhlenbeck labs in chemistry, and the Gold and Yarus labs in biology.

What caught my interest about the T4 introns wasn't so much the chemistry of their catalysis, it was more the biology of why they were in T4 at all. Why would a highly streamlined bacteriophage genome keep three large introns around? My PhD advisor Larry Gold and my co-mentor David Shub had concocted an idea for a possible regulatory function for the T4 introns. The idea was based on the fact that the mechanism of group I splicing requires an exogenous guanosine, and the fact that group I introns seem to occur preferentially in genes having to do with nucleotide metabolism, GTP consumption, and/or nucleotide-like cofactors. For example, two of the three T4 introns were in genes encoding key enzymes in deoxynucleotide synthesis: *td*, the gene for thymidylate synthase, and *nrdB*, one of the two subunits of the aerobic ribonucleotide reductase. Larry and David hypothesized that these introns were regulatory, with splicing rates responsive to intracellular small molecule concentrations—perhaps GTP itself.

The direct prediction of the Gold/Shub model is that if you constructed an intronless phage, it would be impaired for growth under conditions where it needed to down-regulate *td* and *nrdB*. However, two years of my work failed to show any mutant phenotype for the precise triple intron deletion. Indirectly, their idea also made me very interested in searching for more group I introns, not only in T4 but also in other organisms, because you'd expect to see that they too would be in key genes for DNA synthesis, nucleotide metabolism, or GTP consumption.[1]

It seemed to me that it ought to be feasible to just search DNA sequences computationally for new group I introns. The 169 kilobase genome sequence of phage T4 was then nearing completion, thanks to the efforts of Betty Kutter and others. The first fast sequence homology searching programs had begun to appear (Pearson and Lipman's FASTA in 1988, followed quickly by Altschul, Gish, Miller, Myers, and Lipman's first version of BLAST in 1990), and sequence databases were starting to grow quickly. The power of identifying homologs in sequence databases by computational methods was in the air.

My problem was that group I introns generally don't share much similarity in their linear sequence. Programs like

[1]It turned out, from work by Marlene Belfort and others, that the reason T4 has these group I introns is that it can't get rid of them. Many group I introns are mobile elements, encoding "homing" DNA endonucleases that catalyze an efficient directed gene conversion. For introns with active homing endonucleases, mixed infections of intron-plus and intron-minus phage are converted to almost all intron-plus progeny. Nonetheless, the striking and unexplained preference of group I introns for certain host genes continues to follow the Gold/Shub idea. For example, in 1990, the *B. subtilis* phage SPO1 was found to have a group I intron in its DNA polymerase gene. In 1996, the third T4 intron-containing gene, *sunY*, was found to encode the T4 anaerobic ribonucleotide reductase, and has now been renamed *nrdD*.

Corresponding author: sean@eddylab.org
Article and publication date are at http://www.rnajournal.org/cgi/doi/10.1261/rna.050484.115. Freely available online through the *RNA* Open Access option.

BLAST didn't do a good job of finding them. Group I introns do, however, have distinctive conserved secondary structures, easily recognized by eye, at least by gurus like Eric Westhof and Francois Michel, and I got fairly good at it too. For a problem so readily solved by eye (and this distinguishes comparative analysis of RNA secondary structure from, say, protein structure prediction), it seemed that there ought to be computer methods for recognizing both sequence and structure similarity in RNAs. There had been some work in the area, including pioneering work by Daniel Gautheret and Thomas Dandekar, but the available methods were relatively simple pattern matching programs that worked all right for some small RNA structures, but not for complex structures like a group I intron. I wanted the equivalent of BLAST alignment scores, but for RNA secondary structure.

I was doubly fortunate to be at Boulder. Boulder was not just a mecca for RNA research, but also for computational biology, well before computational biology was a thing. Gene Myers (co-developer of BLAST) and David Haussler had come through Boulder as PhD students, and Gary Stormo was on the faculty. I played basketball with Gary on Tuesday mornings, so I asked him. Gary told me that general methods for RNA structure/sequence similarity search were still an open problem. My reaction to this was along the lines of "come on, how hard could this be?" I started learning about computational RNA folding algorithms from reading Michael Zuker's papers, which Gary recommended to me. I didn't get anywhere. The problem looked too hard.

In the last days that I was in Boulder, before I left for a postdoc at the MRC Laboratory in Molecular Biology in the UK, Stormo gave me a preprint of a new paper from David Haussler's postdoc Anders Krogh. The Krogh/Haussler preprint introduced a method called "hidden Markov models" (HMMs) for formalizing linear sequence alignment methods. Gary told me, this manuscript is going to be a big deal. I duly put it in my backpack and didn't read it. Soon after I arrived in Cambridge, Richard Durbin (who later became my postdoctoral mentor, after I committed to switching to computational biology) handed me another copy of the same Krogh/Haussler preprint and said, this manuscript is going to be a big deal. I decided that ok, maybe I should read it.

Both the mathematics and the writing in Krogh's paper were beautiful and clear, clear enough for a biologist like me to understand. HMMs provide a general way of thinking about probabilistic sequence matching problems. Like standard sequence alignment in programs like BLAST, an HMM compares two sequences by having one "state" representing each residue in your query sequence (or consensus profile) to one residue in your target sequence at a time, growing an alignment from left to right. Inspired by Krogh's paper, one fall weekend afternoon in Cambridge I started thinking and doodling in my notebook. What if a "state" represented a base pair? What if it corresponded to *two* residues in the target sequence you were aligning to, so you had a three-way alignment of one model state to two target sequence residues? What if you did the alignment from inside out, adding one or two residues at a time to either or both sides of a growing alignment, as opposed to left to right, one residue at a time? (This inside-out, base-pair-adding style of "dynamic programming" algorithm for RNA was already familiar to me from the way that the Zuker RNA folding algorithm works.) Suddenly I realized I'd solved the problem! You could make an HMM-like model of a structural RNA where your model was a binary tree (reflecting the base-pairing consensus structure of an RNA) instead of a line (a consensus primary sequence, in the Krogh/Haussler profile HMMs). Your model would take into account *both* sequence and structure conservation scoring in a mathematically consistent way that was the natural extension of BLAST scores to RNA alignment. An efficient algorithm existed for computing optimal RNA structural alignments to genome sequences. After a nervous week of double-checking that my algorithms were indeed correct, I showed my idea to Durbin. Durbin immediately said, well, sure, that's a stochastic context-free grammar.

Stochastic context-free grammars (SCFGs) are the next level up from HMMs in a hierarchy of "formal grammar" methods used to model symbol strings. They were well known in the signal processing, speech recognition, and linguistics communities—though not to me. SCFGs had seen some limited application in speech modeling, but human language doesn't tend to have many strong nested pairwise correlations, which is what SCFGs excel at modeling. In contrast, the base pairs of a non-pseudoknotted RNA secondary structure are *all* nested pairwise interactions, perfectly suited to SCFG methods. Later I took perverse pleasure in giving talks in speech engineering groups, describing the RNA structure/sequence alignment problem as an introductory teaser. Old hands in SCFGs would be vibrating in their seats. Finally, a problem that the technology was suited to solving!

Of course, the fact that SCFGs were a well-known technology looking for an application in computer engineering departments meant that some real computer scientist close enough to RNA biology was bound to see the connection too, without needing to reinvent the wheel the way I had. Sure enough, I soon heard that Yasu Sakakibara, later to become a friend of mine, was already working on SCFGs for RNA sequence/structure alignment within the Haussler lab at Santa Cruz. Yasu and I coordinated the submission of our two manuscripts in 1994.

There is a big difference between the sort of proof of principle that suffices as a publishable result in computational biology, as opposed to what it takes to make a truly useful software tool. The algorithms that Yasu and I described were correct, but prohibitively expensive in both memory and time. Memory requirements limited us to structural RNAs of not much larger than 100–150 nt. Group I introns were well out of reach. Time requirements meant that even for small RNAs, we could only afford to search small amounts of target sequence, not whole sequence databases.

We made the most of what we had, for smaller structural RNAs. I produced a first implementation of general profile SCFG techniques, called COVE. Todd Lowe, my first graduate student at Washington University, developed a program for finding tRNA genes by putting two existing tRNA search programs, tRNAscan (from Gwennaele Fichant and Christian Burks) and EufindtRNA (from Giulio Pavesi's group), as fast filters in front of a slow but powerful profile SCFG of tRNA consensus. Todd's tRNAscan-SE program is still in use today for genome annotation.

Slowly and steadily over time, we whittled down the computational problems. In 2002, I found a new algorithm that solved the memory problem. I implemented that as the heart of a new software suite called Infernal, which replaced COVE. The new memory-efficient algorithm cleared the way for using these methods more widely, because although you can't do much about not having enough memory, you can brute force your way around a speed problem in various ways.

In 2003, Sam Griffiths-Jones, Alex Bateman and colleagues at the Sanger Centre in Cambridge released the first version of the Rfam structural RNA families database, using that early version of Infernal. Today Rfam is one of the most comprehensive databases of RNA families, widely used for genome annotation of structural RNA homologs. Rfam worked around the speed issue by using low-stringency BLAST searches as a pre-filter to identify subsequences for subsequent Infernal scanning. This was less than ideal, but was the best compromise at the time.

With the memory limitation solved, then speed advances started to come, piece by piece. In 2004, Zasha Weinberg, a PhD student in Larry Ruzzo's lab in computer science in Seattle, started publishing acceleration algorithms for Infernal. Zasha then moved to a postdoc in Ron Breaker's group at Yale and started using Infernal routinely as part of the Breaker lab's remarkable run of using bioinformatics to prospect for new riboswitches. Eric Nawrocki joined my lab as a PhD student, also in 2004, and his first project was another algorithmic acceleration that synergized with Zasha's work. Eric took responsibility for the growing suite of acceleration methods in the Infernal code, and he soon became the overlord of Infernal development, and a co-czar of Rfam as well. Meanwhile I started concentrating on the problem of accelerating profile HMM searches (the Krogh/Haussler methods, whose use were also limited by their computational requirements) in my HMMER software, working with Michael Farrar, a computer systems engineer in the telecommunications industry whose spare-time hobby was accelerating sequence alignment algorithms. After algorithms enabling 100- to 1000-fold HMMER accelerations were released in 2011, based on Farrar's work, Eric incorporated the same techniques into Infernal.

This long arc of work reached a big landmark this year. In its most recent release, Rfam has discarded its BLAST filters. For the first time, the Rfam team is using native Infernal searches for all 2450 structural RNA families in the database. At long last, SCFG search methods for RNA structure are now fast and efficient enough that they can be used systematically, for basically any RNA, on essentially any sequence database. Eric's current version of Infernal is about 10,000-fold faster than my original 2002 version. Instead of taking cpu-*centuries* to search the nonredundant sequence database for an RNA structure the size of a group I intron, now that search can be done in a couple of cpu-*days*. Still expensive, but entirely feasible.

For the 2015 Rfam release paper, one of the computational experiments that Eric did was to compare the old BLAST filter pipeline to the new native Infernal searches. We expected that using the BLAST filters had been compromising our detection sensitivity for RNAs that have a strongly conserved secondary structure but weakly conserved sequence. Eric was looking for Rfam families with the biggest increase in number of putative detected homologs. I was looking over his results and there at the top of Eric's ranked list: over 10,000 putative new group I introns detected by the native Infernal search but not with Rfam's previous pipeline.

It feels like we've come full circle. It's been so long that no one in my lab works on group I introns any more. We had to go so far down a rabbit hole of algorithm development and computer engineering to make the tool practical, that we never went back and used our own software for the original reason I wanted to have it. But now as I write this, Eric and Tom Jones in my lab are happily sifting through all these new putative group I introns—at long last, the results of the search that I wanted to do as a PhD student in Larry's lab 20 years ago.