GUI Frontend for UNAFold

October 2008

Contents

1 Introduction

As with any project or task that uses resources, the ability to model the folding of designed ribonucleic acid (RNA) is desired for its potential to save time, money, and supplies. Reasonably accurate secondary structure prediction could allow the examination of not only the structure of the sequence in question, but also how it interacts with other sequences that may be present in its target environment (e.g. a cell). By carrying out modelling, it may be possible to avoid problems before the RNA is even synthesised, or any lab time or supplies has been used on it.

2 Goals

The initial goals where straightforward: to improve the graphic user interface (GUI) of a RNA secondary structure prediction program such that it would be easier for the end-user, thus increasing the amount of modelling being done. Also desired was a way to allow the program to process the interactions of more than two sequences simultaneously. This was viewed as an item that would be a great help for the examination of behaviour of RNA logic gates, as one could model the interactions between different inputs and gates at the same time.

3 RNA Secondary Structure Prediction Software

3.1 Currently Available Software

The original plan was to use a program called RNAStructure [1] as the starting point. This piece of software is said to have achieved a very high (80%) success rate for determining the secondary structures of RNA sequences. Unfortunately, this software was not very easy to work with from a development perspective, and an alternative was found in the UNAFold program [2]. UNAFold is the older of they two programs, but seemed to be easier to work with and could be compiled on a variety of different operating systems, whereas the RNAStructure is a Windows program.

In addition to these differences, it should be mentioned that while RNAStructure provides its own GUI, UNAFold is a command line program, meaning that its run through a series of text commands and does not make use of graphics or mouse clicks.

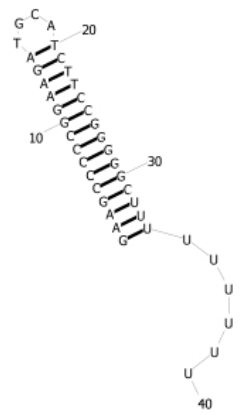

Figure 1: Prediction of a Hairpin Structure

3.2 Comparison between Pieces of RNA Folding Software

Since the RNAStructure program is newer and has better accuracy than the older UNAFold, comparisons where made to determine if so much accuracy was being lost as to make the use of UNAFold worse than staying with the presently used RNAStructure. For smaller basic structures, such as a basic hairpin (Fig. 1), both programs provided identical structure outputs, and only had a slight difference in free energy estimates.

For more complicated single sequence folds, the programs differed slightly in both structure and free energy estimate. Fig. 2 shows the structure predicted using the UNAFold program, while Fig. 3 shows the structure predicted by RNAStructure.

The two sequence fold predictions were where the programs really started to diverge. Figures 5 and 4 show the output of UNAFold and RNAStructure (respectively) when the programs were provided with two copies of the TA11In sequence.

4 GUI Development

It was decided to try and develop a GUI for UNAFold that would meet some of the initial goals laid out in Sec. 2, as this seemed to have a better chance of getting something accomplished.

4.1 Open Source

In keeping with the spirit of the open source nature of International Genetically Engineered Machine (iGEM) competition, an effort was made to make use of programs and tools that are freely available on the internet. For this reason, the current iteration of this user interface

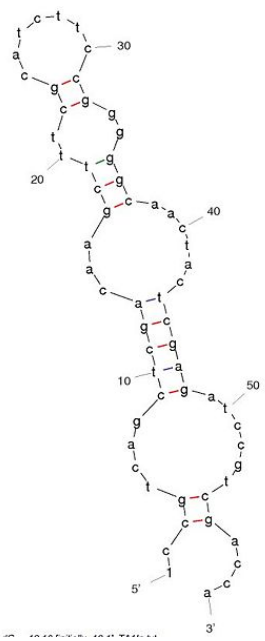

Figure 2: TA1In Sequence: UNAFold

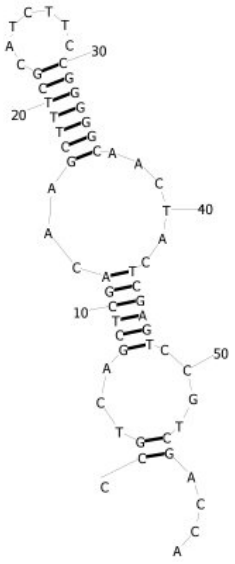

Figure 3: TA1In Sequence: RNAStructure

was designed to run on the Ubuntu [3] operating system, one of the easier to use Linux distributions. The interface code was written in Python [4], with the GUI being designed using wxGlade [5], which makes use of the wxPython [6] libraries. At the time of writing, all of these items, including the operating system, are available for free on the internet with open source licenses. To use this GUI in its provided form, Python and wxPython should be installed on an Ubuntu system. The GUI also makes use of a small program called evince to view the postscript files output from the plotting utility, but this program is installed by default in Ubuntu, so no action should be necessary with that.

Due to the cross-platform nature of these tools, it should be relatively easy to change them to work on other operating systems (e.g. Windows). The current source code is provided in Sec. 5, and more current versions (if any) will be sent upon request.

4.2 Other Required Installations

In addition to the software listed above, the other installations necessary are, of course, the actual UNAFold program, and mfold_utils, which is used for the visualisation of the predicted fold. At the time of writing, both are available at
`http://dinamelt.bioinfo.rpi.edu/download.php`.

4.3 Method

4.3.1 Graphic User Interface

As previously mentioned, the GUI for this program was built using the wxGlade designer tool [5]. This allowed for fast development of the parent window and necessary child dialogs. Using this tool, future extensions should be made simpler. The program consists of different child dialog boxes that reside inside the parent window. Single sequence folds are handled in a different dialog than multi-sequence, and there can be any number of these dialog boxes open at a time. All dialog boxes contain text fields where the sequence(s) can be entered, and a button to execute the fold. The sequences can have any number of spaces or carriage returns, as the interface will remove them before passing the sequence on. Currently, the interface does nothing about the user entering strange values (i.e. letters that are not G,C,T,A or U),

4.3.2 Single Folds

On the button click event, the program removes the whitespace from the sequence and then saves it to a temporary file. This temporary file name is then strung together with predefined strings to form a command which will call the UNAFold program. This command is then passed to the system, which executes it. The output filenames are known in advance, from the name of the temporary file and the naming conventions used in UNAFold, so this name is then passed to the sir_graph utility via a system command. This program provides a

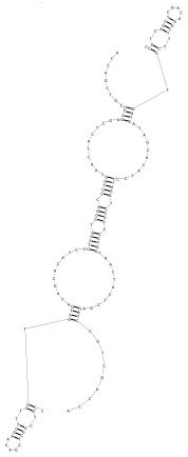

Figure 4: TA1In/TA1In Sequence: RNAStructure

Figure 5: TA1In/TA1In Sequence: UNAFold

visualisation of the secondary structure (as a postscript file), which is then displayed with evince via another system command. Since all the filenames where known throughout the process, the program then deletes most of the intermediate files, and renames the final output to reflect the sequence that was folded.

4.3.3 Multiple Sequence Folding

For two sequences, the interface operates in much the same way that it does with one sequence, except using two temporary files etc. instead of one. This is because UNAFold supports the folding of two sequences. When more than two sequences are to be folded together, the interface takes advantage of some of the quirks of the UNAFold to put all the folds into the process at the same time. First the sequences are stripped of their whitespace, then attached together with another small string of characters. By default this has been made "III". The interface then creates a rule file, which is supported by UNAFold. In this rule file, the program places rules prohibiting UNAFold from bonding anything to the locations occupied by the "III" strings. The result of this is a single sequence containing subsequences attached together by small strings of nonsense, unbondable bases. It is important to note, that for some reason or other, if the rules prohibiting bonding, UNAFold will bond things to these nonsense bases. After the creation of the new sequence and rule file, the process completes by going through the same steps as was done in a single sequence fold.

4.4 Result

Screenshots of the resulting GUI are shown in Fig. 6 and Fig. 7. Unfortunately, the GUI did not see much use by end-users, so it is difficult to say if the system provided is actually preferable to the RNAStructure program. It is not known whether or not there is any merit in the multifold portion of the program. Due to the fact that UNAFold had to be "tricked" into folding a sequence like that, it is suspected that assumptions used in the prediction of RNA secondary structure have broken down with the addition of these sequences and therefore the multifold aspect of the GUI is suspected to be very inaccurate.

4.5 Future Development

Future improvements to this interface include improving the measures that prevent users from providing nonsensical inputs. Depending on feedback from end users, some of the GUI behaviour may have to be changed (e.g. text field scroll bars, updating text fields with sequences after whitespace removal). The ability to change the case (i.e. upper or lower) of the input sequence may also be desired for the future. Also, the ability to draw the visualisation of the sequence and display it rather than rely on external programs may be a potentially desired development. This may be achievable through the use of one of the graphics packages available for Python. Finally, perhaps packaging the whole system, as well as porting the code to other operating systems would be desirable as a way to improve distribution.

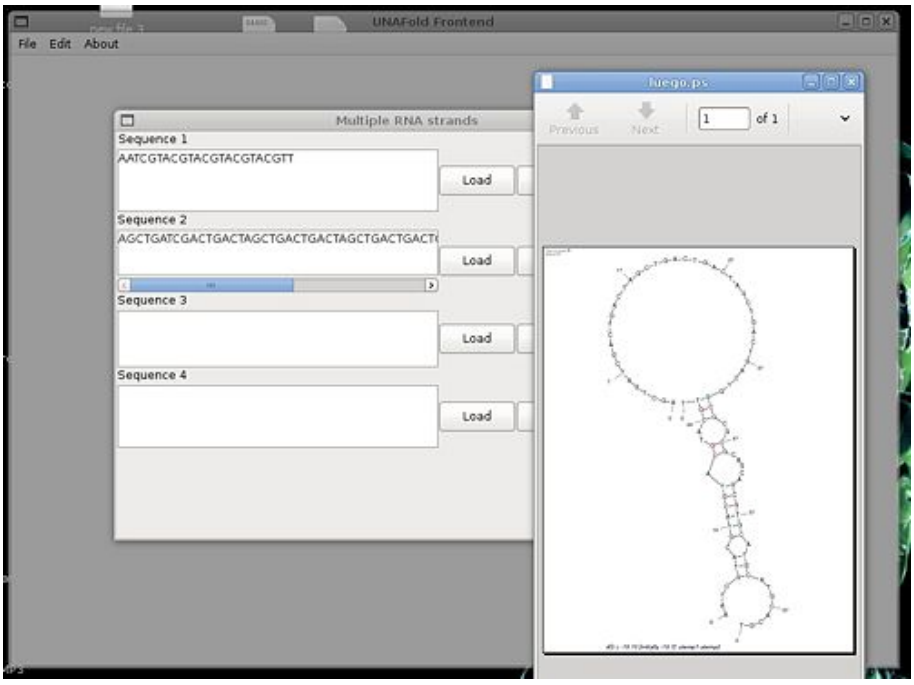

Figure 6: Screenshot of The Developed GUI Interface

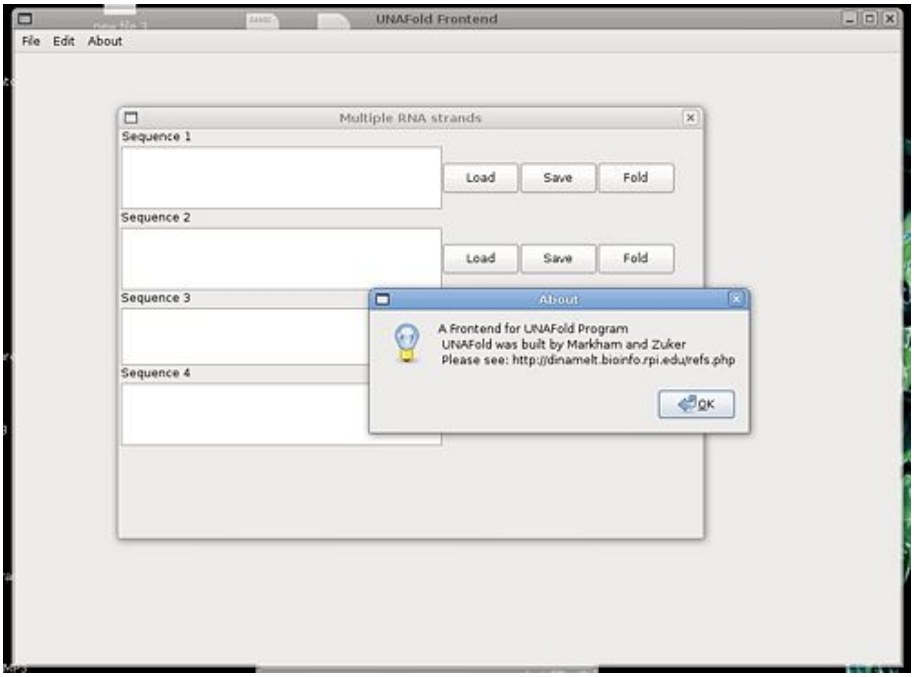

Figure 7: Another Screenshot of The Developed GUI Interface

References

[1] D.H. Mathews, M.D. Disney, J.L. Childs, S.J. Schroeder M. Zuker, and D.H Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences, USA*, II:7287–7292, 2004.

[2] M. Markham, N. R. & Zuker. Unafold: software for nucleic acid folding and hybriziation. *Methods in Molecular Biology*, II:331, 2008.

[3] Canonical Ltd. Ubuntu. Ubuntu Website, October 2008. `http://www.ubuntu.com/`.

[4] G. van Rossum et al. Python Language Website, October 2008. `http://www.python.org`.

[5] Alberto Griggio. wxGlade Website, September 2008. `http://wxglade.sourceforge.net`.

[6] Julian Smart, Robert Roebling, Vadim Zeitlin, Robin Dunn, and et al. wxPython Website, September 2008. `http://wxpython.org/`.

5 Appendix: Complete Source Code

The source code for this program is included below. Due to the way Python works, filenames are important for the modules, unless you change the `import` statements near the top of the files. All these files should be located in the same directory and the `UNAFoldFrontend.py` file is the one that should be run.

5.1 UNAFoldFrontend.py

```
#!/usr/bin/env python
# -*- coding: iso-8859-1 -*-
# generated by wxGlade 0.5 on Thu May 08 11:13:32 2008

import wx
import os
import singFold
import multiFold

ID_NEW = 001
ID_OPEN = 002
ID_EXIT = 003
ID_ABOUT = 004
ID_MULTIRNA =005

class MyDialog3(wx.Dialog):
    def __init__(self, *args, **kwds):
        # begin wxGlade: MyDialog3.__init__
        kwds["style"] = wx.DEFAULT_DIALOG_STYLE|wx.MINIMIZE_BOX
        wx.Dialog.__init__(self, *args, **kwds)
        self.label_1 = wx.StaticText(self, -1, "Sequence 1", style=wx.ALIGN_CENTRE)
        self.text_ctrl_seq1 = wx.TextCtrl(self, -1, "",\
style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_LINEWRAP|wx.TE_WORDWRAP)
        self.load_seq1 = wx.Button(self, -1, "Load")
        self.save_seq1 = wx.Button(self, -1, "Save")
        self.foldSeq1 = wx.Button(self, -1, "Fold")
        self.label_2 = wx.StaticText(self, -1, "Sequence 2")
        self.text_ctrl_seq2 = wx.TextCtrl(self, -1, "",\
 style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_LINEWRAP|wx.TE_WORDWRAP)
        self.load_seq2 = wx.Button(self, -1, "Load")
        self.save_seq2 = wx.Button(self, -1, "Save")
        self.foldSeq2 = wx.Button(self, -1, "Fold")
        self.label_3 = wx.StaticText(self, -1, "Sequence 3")
        self.text_ctrl_seq3 = wx.TextCtrl(self, -1, "",\
 style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_LINEWRAP|wx.TE_WORDWRAP)
        self.load_seq3 = wx.Button(self, -1, "Load")
        self.save_seq3 = wx.Button(self, -1, "Save")
        self.foldSeq3 = wx.Button(self, -1, "Fold")
        self.label_4 = wx.StaticText(self, -1, "Sequence 4")
        self.text_ctrl_seq4 = wx.TextCtrl(self, -1, "",\
 style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_LINEWRAP|wx.TE_WORDWRAP)
        self.load_seq4 = wx.Button(self, -1, "Load")
        self.save_seq4 = wx.Button(self, -1, "Save")
        self.foldSeq4 = wx.Button(self, -1, "Fold")

        self.__set_properties()
```

```
        self.__do_layout()

        self.Bind(wx.EVT_BUTTON, self.OnLoadSeq1, self.load_seq1)
        self.Bind(wx.EVT_BUTTON, self.OnSaveSeq1, self.save_seq1)
        self.Bind(wx.EVT_BUTTON, self.OnMultiFold, self.foldSeq1)
        self.Bind(wx.EVT_BUTTON, self.OnLoadSeq2, self.load_seq2)
        self.Bind(wx.EVT_BUTTON, self.OnSaveSeq2, self.save_seq2)
        self.Bind(wx.EVT_BUTTON, self.OnMultiFold, self.foldSeq2)
        self.Bind(wx.EVT_BUTTON, self.OnLoadSeq3, self.load_seq3)
        self.Bind(wx.EVT_BUTTON, self.OnSaveSeq3, self.save_seq3)
        self.Bind(wx.EVT_BUTTON, self.OnMultiFold, self.foldSeq3)
        self.Bind(wx.EVT_BUTTON, self.OnLoadSeq4, self.load_seq4)
        self.Bind(wx.EVT_BUTTON, self.OnSaveSeq4, self.save_seq4)
        self.Bind(wx.EVT_BUTTON, self.OnMultiFold, self.foldSeq4)
        # end wxGlade

    def __set_properties(self):
        # begin wxGlade: MyDialog3.__set_properties
        self.SetTitle("Multiple RNA strands")
        self.SetSize((634, 444))
        self.text_ctrl_seq1.SetMinSize((351, 68))
        self.text_ctrl_seq2.SetMinSize((351, 68))
        self.text_ctrl_seq3.SetMinSize((351, 62))
        self.text_ctrl_seq4.SetMinSize((351, 68))
        # end wxGlade

    def __do_layout(self):
        # begin wxGlade: MyDialog3.__do_layout
        grid_sizer_1 = wx.FlexGridSizer(8, 2, 0, 0)
        sizer_8 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_7 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_6 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_5 = wx.BoxSizer(wx.HORIZONTAL)
        grid_sizer_1.Add(self.label_1, 0, 0, 0)
        grid_sizer_1.Add((20, 20), 0, wx.EXPAND, 0)
        grid_sizer_1.Add(self.text_ctrl_seq1, 0, 0, 0)
        sizer_5.Add(self.load_seq1, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_5.Add(self.save_seq1, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_5.Add(self.foldSeq1, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        grid_sizer_1.Add(sizer_5, 1, wx.EXPAND, 0)
        grid_sizer_1.Add(self.label_2, 0, 0, 0)
        grid_sizer_1.Add((20, 20), 0, wx.EXPAND, 0)
        grid_sizer_1.Add(self.text_ctrl_seq2, 0, 0, 0)
        sizer_6.Add(self.load_seq2, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_6.Add(self.save_seq2, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_6.Add(self.foldSeq2, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        grid_sizer_1.Add(sizer_6, 1, wx.EXPAND, 0)
        grid_sizer_1.Add(self.label_3, 0, 0, 0)
        grid_sizer_1.Add((20, 20), 0, wx.EXPAND, 0)
        grid_sizer_1.Add(self.text_ctrl_seq3, 0, 0, 0)
        sizer_7.Add(self.load_seq3, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_7.Add(self.save_seq3, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_7.Add(self.foldSeq3, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        grid_sizer_1.Add(sizer_7, 1, wx.EXPAND, 0)
```

```
        grid_sizer_1.Add(self.label_4, 0, 0, 0)
        grid_sizer_1.Add((20, 20), 0, wx.EXPAND, 0)
        grid_sizer_1.Add(self.text_ctrl_seq4, 0, 0, 0)
        sizer_8.Add(self.load_seq4, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_8.Add(self.save_seq4, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_8.Add(self.foldSeq4, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        grid_sizer_1.Add(sizer_8, 1, wx.EXPAND, 0)
        self.SetSizer(grid_sizer_1)
        self.Layout()
        # end wxGlade

    def OnLoadSeq1(self, event): # wxGlade: MyDialog3.<event_handler>
        self.dirname = ''
        dlg = wx.FileDialog(self, "Choose a file", self.dirname, "",\
"*.seq", wx.OPEN)
        if dlg.ShowModal() == wx.ID_OK:
            self.filename=dlg.GetFilename()
            self.dirname=dlg.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'r')

            self.text_ctrl_seq1.SetValue(f.read())
            f.close()
            dlg.Destroy()




    def OnSaveSeq1(self, event): # wxGlade: MyDialog3.<event_handler>
        self.dirname = ''
        dlgSaveNew = wx.FileDialog(self, "Choose a file", self.dirname, "",\
"*.seq", wx.OPEN)
        if dlgSaveNew.ShowModal() == wx.ID_OK:
            self.filename=dlgSaveNew.GetFilename()
            self.dirname=dlgSaveNew.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'w')
            seq = self.text_ctrl_seq1.Value
            f.write(seq)
            f.close()
            dlgSaveNew.Destroy()

    def OnMultiFold(self, event): # wxGlade: MyDialog3.<event_handler>

        maxSeq = 2
        filledSeq = 0

        if self.text_ctrl_seq1.Value != '':
            filledSeq = filledSeq + 1
            print "one filled"
        if self.text_ctrl_seq2.Value != '':
            filledSeq = filledSeq + 1
            print "two filled"
        if self.text_ctrl_seq3.Value != '':
```

```
            filledSeq = filledSeq + 1
            print "three filled"
        if self.text_ctrl_seq4.Value != '':
            filledSeq = filledSeq + 1
            print "four filled"

        print filledSeq

        if filledSeq == 0:
            a = wx.MessageDialog(self, "No Sequences", "Error", wx.OK)
            a.ShowModal()
            a.Destroy()
            event.Skip()

        if filledSeq ==1:
            a = wx.MessageDialog(self, "For one sequence, please use\
monomolecular menu", "Error", wx.OK)
            a.ShowModal()
            a.Destroy()
            event.Skip()



        self.dirname = ''

        #get a working directory
        dlgSaveNew = wx.FileDialog(self, "Choose a file", self.dirname, "",\
"*.ps", wx.OPEN)
        if dlgSaveNew.ShowModal() == wx.ID_OK:
            self.filename=dlgSaveNew.GetFilename()
            self.dirname=dlgSaveNew.GetDirectory()
            #f=open(os.path.join(self.dirname,self.filename),'w')



            dlgSaveNew.Destroy()

        #collect text control contents

        a = self.text_ctrl_seq1.Value
        b = self.text_ctrl_seq2.Value
        c = self.text_ctrl_seq3.Value
        d = self.text_ctrl_seq4.Value

        f = self.filename
        g = self.dirname


        if filledSeq > 2:

            multiFold.attemptedLotsOfStrands(a,b,c,d,f,g)

            a = wx.MessageDialog(self, "Over two seq input. Don't be expecting\
miracles.  Sorry man.", "Error", wx.OK)
```

```
            a.ShowModal()
            a.Destroy()

        else:
            multiFold.multFold(a,b,c,d,f,g)


    def OnLoadSeq2(self, event): # wxGlade: MyDialog3.<event_handler>
        self.dirname = ''
        dlg = wx.FileDialog(self, "Choose a file", self.dirname, "", "*.seq",\
wx.OPEN)
        if dlg.ShowModal() == wx.ID_OK:
            self.filename=dlg.GetFilename()
            self.dirname=dlg.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'r')

            self.text_ctrl_seq2.SetValue(f.read())
            f.close()
            dlg.Destroy()

    def OnSaveSeq2(self, event): # wxGlade: MyDialog3.<event_handler>
        self.dirname = ''
        dlgSaveNew = wx.FileDialog(self, "Choose a file", self.dirname, "",\
"*.seq", wx.OPEN)
        if dlgSaveNew.ShowModal() == wx.ID_OK:
            self.filename=dlgSaveNew.GetFilename()
            self.dirname=dlgSaveNew.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'w')
            seq = self.text_ctrl_seq2.Value
            f.write(seq)
            f.close()
            dlgSaveNew.Destroy()

    def OnLoadSeq3(self, event): # wxGlade: MyDialog3.<event_handler>
        self.dirname = ''
        dlg = wx.FileDialog(self, "Choose a file", self.dirname, "", "*.seq",\
wx.OPEN)
        if dlg.ShowModal() == wx.ID_OK:
            self.filename=dlg.GetFilename()
            self.dirname=dlg.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'r')

            self.text_ctrl_seq3.SetValue(f.read())
            f.close()
            dlg.Destroy()
    def OnSaveSeq3(self, event): # wxGlade: MyDialog3.<event_handler>
        self.dirname = ''
        dlgSaveNew = wx.FileDialog(self, "Choose a file", self.dirname, "",\
"*.seq", wx.OPEN)
        if dlgSaveNew.ShowModal() == wx.ID_OK:
            self.filename=dlgSaveNew.GetFilename()
            self.dirname=dlgSaveNew.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'w')
            seq = self.text_ctrl_seq3.Value
```

```
            f.write(seq)
            f.close()
            dlgSaveNew.Destroy()

    def OnLoadSeq4(self, event): # wxGlade: MyDialog3.<event_handler>
        self.dirname = ''
        dlg = wx.FileDialog(self, "Choose a file", self.dirname, "", "*.seq",\
 wx.OPEN)
        if dlg.ShowModal() == wx.ID_OK:
            self.filename=dlg.GetFilename()
            self.dirname=dlg.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'r')

            self.text_ctrl_seq4.SetValue(f.read())
            f.close()
            dlg.Destroy()

    def OnSaveSeq4(self, event): # wxGlade: MyDialog3.<event_handler>
        self.dirname = ''
        dlgSaveNew = wx.FileDialog(self, "Choose a file", self.dirname, "",\
 "*.seq", wx.OPEN)
        if dlgSaveNew.ShowModal() == wx.ID_OK:
            self.filename=dlgSaveNew.GetFilename()
            self.dirname=dlgSaveNew.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'w')
            seq = self.text_ctrl_seq4.Value
            f.write(seq)
            f.close()
            dlgSaveNew.Destroy()

# end of class MyDialog3


class MyDialog2(wx.Dialog):
    def __init__(self, *args, **kwds):
        # begin wxGlade: MyDialog2.__init__
        kwds["style"] = wx.DEFAULT_DIALOG_STYLE
        wx.Dialog.__init__(self, *args, **kwds)
        self.text_ctrl_2 = wx.TextCtrl(self, -1, "", style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_LINEWRAP|wx.TE
        self.button_2 = wx.Button(self, -1, "Fold")
        self.button_3 = wx.Button(self, -1, "Save")

        self.__set_properties()
        self.__do_layout()

        self.Bind(wx.EVT_BUTTON, self.OnNewFold, self.button_2)
        self.Bind(wx.EVT_BUTTON, self.OnNewSave, self.button_3)
        # end wxGlade

    def __set_properties(self):
        # begin wxGlade: MyDialog2.__set_properties
        self.SetTitle("New RNA Sequence")
        self.text_ctrl_2.SetMinSize((883, 150))
        self.button_2.Enable(False)
```

```
        # end wxGlade

    def __do_layout(self):
        # begin wxGlade: MyDialog2.__do_layout
        sizer_2 = wx.BoxSizer(wx.VERTICAL)
        sizer_4 = wx.BoxSizer(wx.HORIZONTAL)
        sizer_2.Add(self.text_ctrl_2, 0, wx.EXPAND, 0)
        sizer_4.Add(self.button_2, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_4.Add((600, 20), 0, 0, 0)
        sizer_4.Add(self.button_3, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_2.Add(sizer_4, 50, 0, 0)
        self.SetSizer(sizer_2)
        sizer_2.Fit(self)
        self.Layout()
        # end wxGlade

    def OnNewFold(self, event): # wxGlade: MyDialog2.<event_handler>
        ##seq = self.text_ctrl_2.Value
        self.dirname = ''

        dlgSaveNew = wx.FileDialog(self, "Choose an Output file", self.dirname,\
"", "*.ct", wx.OPEN)
        if dlgSaveNew.ShowModal() == wx.ID_OK:
            self.filename=dlgSaveNew.GetFilename()
            self.dirname=dlgSaveNew.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'w')
            g = self.filename
            #seq = self.text_ctrl_2.Value
            #f.write(seq)

            #f.close()
            dlgSaveNew.Destroy()

        seqStripped =  singFold.oneFold(self.text_ctrl_2.Value,f, g)
        self.text_ctrl_2.SetValue(seqStripped)


    def OnNewSave(self, event): # wxGlade: MyDialog2.<event_handler>
        self.dirname = ''
        dlgSaveNew = wx.FileDialog(self, "Choose a file", self.dirname, "",\
"*.seq", wx.OPEN)
        if dlgSaveNew.ShowModal() == wx.ID_OK:
            self.filename=dlgSaveNew.GetFilename()
            self.dirname=dlgSaveNew.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'w')
            seq = self.text_ctrl_2.Value
            f.write(seq)
            #dlgOpen.text_ctrl_2.SetValue(f.read())
            f.close()
            dlgSaveNew.Destroy()
            #strDialogCount = strDialogCount + "1"
            self.button_2.Enable(True)
            g = self.filename
        #print "Event handler `OnNewSave' not implemented!"
```

```
        #event.Skip()

# end of class MyDialog2


class MyDialog(wx.Dialog):
    def __init__(self, *args, **kwds):
        # begin wxGlade: MyDialog.__init__
        kwds["style"] = wx.DEFAULT_DIALOG_STYLE|wx.RESIZE_BORDER|wx.MINIMIZE_BOX|wx.THICK_FRAME
        wx.Dialog.__init__(self, *args, **kwds)
        self.text_ctrl_1 = wx.TextCtrl(self, -1, "", style=wx.TE_MULTILINE|wx.HSCROLL|wx.TE_LINEWRAP|wx.TE
        self.button_1 = wx.Button(self, -1, "Fold")
        self.button_4 = wx.Button(self, -1, "Save")
        self.button_5 = wx.Button(self, -1, "Save As")

        self.__set_properties()
        self.__do_layout()

        self.Bind(wx.EVT_BUTTON, self.OnSingFold, self.button_1)
        self.Bind(wx.EVT_BUTTON, self.OnOpenSave, self.button_4)
        self.Bind(wx.EVT_BUTTON, self.OnOpenSaveAs, self.button_5)
        # end wxGlade

        global strDialogCount
        strDialogCount = "1"

    def __set_properties(self):
        # begin wxGlade: MyDialog.__set_properties
        self.SetTitle("Single RNA Sequence")
        self.text_ctrl_1.SetMinSize((883, 150))
        self.button_4.Enable(False)
        # end wxGlade

    def __do_layout(self):
        # begin wxGlade: MyDialog.__do_layout
        sizer_3 = wx.BoxSizer(wx.VERTICAL)
        sizer_3.Add(self.text_ctrl_1, 0, wx.EXPAND, 0)
        sizer_3.Add(self.button_1, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_3.Add(self.button_4, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        sizer_3.Add(self.button_5, 0, wx.ALIGN_CENTER_HORIZONTAL|wx.ALIGN_CENTER_VERTICAL, 0)
        self.SetSizer(sizer_3)
        sizer_3.Fit(self)
        self.Layout()
        # end wxGlade

    def OnSingFold(self, event): # wxGlade: MyDialog.<event_handler>
        self.dirname = ''

        dlgSaveNew = wx.FileDialog(self, "Choose an Output file", self.dirname,\
 "", "*.ct", wx.OPEN)
        if dlgSaveNew.ShowModal() == wx.ID_OK:
            self.filename=dlgSaveNew.GetFilename()
            self.dirname=dlgSaveNew.GetDirectory()
            dlgSaveNew.Destroy()
```

```
        name = str.split(str(self.filename), '.')
        temp = name[0]
        name = []
        name = temp
        self.filename = name

        name = self.filename
        name = name + ".out"

        f=open(os.path.join(self.dirname,name),'w')

        g = os.path.join(self.dirname,name)
        q=open(g,'w')
        q.write(self.text_ctrl_1.Value)
        q.close()

        singFold.oneFold(self.text_ctrl_1.Value, f, g)
        print g
    def OnOpenSave(self, event): # wxGlade: MyDialog.<event_handler>
        f=open(os.path.join(self.dirname,self.filename),'w')
        seq = self.text_ctrl_1.Value
        f.write(seq)
        f.close()

    def OnOpenSaveAs(self, event): # wxGlade: MyDialog.<event_handler>
        self.dirname = ''
        dlgSaveNew = wx.FileDialog(self, "Choose a file", self.dirname, "",\
 "*.seq", wx.OPEN)
        if dlgSaveNew.ShowModal() == wx.ID_OK:
            self.filename=dlgSaveNew.GetFilename()
            self.dirname=dlgSaveNew.GetDirectory()
            f=open(os.path.join(self.dirname,self.filename),'w')
            seq = self.text_ctrl_1.Value
            f.write(seq)
            f.close()
            dlgSaveNew.Destroy()
            self.button_4.Enable(True)
            g = self.filename
# end of class MyDialog




class MyFrame(wx.Frame):
    def __init__(self, *args, **kwds):
        # begin wxGlade: MyFrame.__init__
        kwds["style"] = wx.DEFAULT_FRAME_STYLE
        wx.Frame.__init__(self, *args, **kwds)

        # Menu Bar
        self.frame_1_menubar = wx.MenuBar()
```

```
        wxglade_tmp_menu = wx.Menu()
        self.NEW = wx.MenuItem(wxglade_tmp_menu, ID_NEW, "New", "",\
wx.ITEM_NORMAL)
        wxglade_tmp_menu.AppendItem(self.NEW)
        self.OPEN = wx.MenuItem(wxglade_tmp_menu, ID_OPEN, "Open", "",\
wx.ITEM_NORMAL)
        wxglade_tmp_menu.AppendItem(self.OPEN)
        wxglade_tmp_menu.AppendSeparator()
        wxglade_tmp_menu.Append(ID_MULTIRNA, "Multiple RNA...", "",\
wx.ITEM_NORMAL)
        wxglade_tmp_menu.AppendSeparator()
        self.EXIT = wx.MenuItem(wxglade_tmp_menu, ID_EXIT, "Exit", "",\
wx.ITEM_NORMAL)
        wxglade_tmp_menu.AppendItem(self.EXIT)
        self.frame_1_menubar.Append(wxglade_tmp_menu, "File")
        wxglade_tmp_menu = wx.Menu()
        self.frame_1_menubar.Append(wxglade_tmp_menu, "Edit")
        wxglade_tmp_menu = wx.Menu()
        self.ABOUT = wx.MenuItem(wxglade_tmp_menu, ID_ABOUT, "About", "",\
wx.ITEM_NORMAL)
        wxglade_tmp_menu.AppendItem(self.ABOUT)
        self.frame_1_menubar.Append(wxglade_tmp_menu, "About")
        self.SetMenuBar(self.frame_1_menubar)
        # Menu Bar end

        self.__set_properties()
        self.__do_layout()

        self.Bind(wx.EVT_MENU, self.OnNew, self.NEW)
        self.Bind(wx.EVT_MENU, self.OnOpen, self.OPEN)
        self.Bind(wx.EVT_MENU, self.OnMultiRNA, id=ID_MULTIRNA)
        self.Bind(wx.EVT_MENU, self.OnExit, self.EXIT)
        self.Bind(wx.EVT_MENU, self.OnAbout, self.ABOUT)
        # end wxGlade

    def __set_properties(self):
        # begin wxGlade: MyFrame.__set_properties
        self.SetTitle("UNAFold Frontend")
        self.SetSize((968, 692))
        # end wxGlade

    def __do_layout(self):
        # begin wxGlade: MyFrame.__do_layout
        sizer_1 = wx.BoxSizer(wx.VERTICAL)
        self.SetSizer(sizer_1)
        self.Layout()
        # end wxGlade




    def OnNew(self, event): # wxGlade: MyFrame.<event_handler>
        dlgNew = MyDialog2(self)
        dlgNew.Show()
```

```
    def OnOpen(self, event): # wxGlade: MyFrame.<event_handler>
        self.dirname = ''
        dlg = wx.FileDialog(self, "Choose a file", self.dirname, "", "*.seq",\
 wx.OPEN)
        if dlg.ShowModal() == wx.ID_OK:
            self.filename=dlg.GetFilename()
            self.dirname=dlg.GetDirectory()
            flName = self.filename
            flDir = self.dirname
            f=open(os.path.join(self.dirname,self.filename),'r')
            dlgOpen = MyDialog(self)
            dlgOpen.text_ctrl_1.SetValue(f.read())
            f.close()
            dlg.Destroy()

            dlgOpen.Show()

    def OnExit(self, event): # wxGlade: MyFrame.<event_handler>
        self.Close(True) #Close frame

    def OnAbout(self, event): # wxGlade: MyFrame.<event_handler>
        a = wx.MessageDialog(self,\
 "A Frontend for UNAFold Program \n UNAFold was built by Markham and Zuker" + \
"\n Please see: http://dinamelt.bioinfo.rpi.edu/refs.php", "About", wx.OK)
        a.ShowModal()
        a.Destroy()

    def OnMultiRNA(self, event): # wxGlade: MyFrame.<event_handler>
        dlgMulti = MyDialog3(self)
        dlgMulti.Show()

# end of class MyFrame


if __name__ == "__main__":
    app = wx.PySimpleApp(0)
    wx.InitAllImageHandlers()
    frame_1 = MyFrame(None, -1, "")

    global intDialogCount
    intDialogCount = 1


    app.SetTopWindow(frame_1)
    frame_1.Show()
    app.MainLoop()
```

5.2 singFold.py

```
#single Folding Module

import string
```

```
import StringIO
import os


def oneFold(seq,f,g):


    f.write(seq)
    f.close()


    commandString = 'UNAFold.pl --max=1 ' + str(g) #+ " " + str(g)
    os.system(commandString)

    commandString = 'rm ' + str(g)
    os.system(commandString)

    commandString = "sir_graph -p " + str(g) + ".ct"
    os.system(commandString)

    commandString ="evince --preview " + str(g) + ".ps &"
    os.system(commandString)

    ##remove old outputs

    commandString = 'rm ' + str(g) + "_*"
    os.system(commandString)

    print "Some intermediate output files deleted...  (normal)"

    print "done"


    return
```

5.3 multiFold.py

```
#multiFold Module

import os

def multFold(a,b,c,d,h,g):

    #a,b,c,d are the sequences to be folded

    #f and g are file/path names


    temp1 = 'atemp1'
    temp2 = 'atemp2'
    temp3 = 'atemp3'
    temp4 = 'atemp4'
```

```
    f=open(temp1,'w')
    f.write(a)
    f.close()

    f=open(temp2,'w')
    f.write(b)
    f.close()



    #execute the fold

    commandString = 'UNAFold.pl --max=1 ' + temp1 + " " + temp2
    os.system(commandString)

    #rename connection table for ease of graphing

    commandString = 'mv atemp1-atemp2_1.ct temp3.ct'
    os.system(commandString)

    #graph the fold

    commandString = 'sir_graph -p temp3.ct'
    os.system(commandString)

    #rename/move the fold visualization

    commandString = 'mv temp3.ps ' + str(g) + "/" + str(h)
    #print commandString
    os.system(commandString)


    #clean up some of the old files

    commandString = 'rm atemp1-atemp2*'
    os.system(commandString)

    commandString = 'rm temp3.ct'
    os.system(commandString)

    #display visualization

    commandString = 'evince --preview ' + str(g) + "/" + str(h) + " &"
    os.system(commandString)

    return


def attemptedLotsOfStrands(a,b,c,d,h,g):

    supportedStrings = 4


    #a,b,c,d are the sequences to be folded
```

```
stringList = [a,b,c,d]
#f and g are file/path names

#print h
#print g

#file names
constraintFile = "constraints.con"
rnaTempFile = "rnaTriQuadTemp"


#string to be folded
foldString = ''

#string to insert between strands of RNA
bufferString = "III"
bufferStringLen  = len(bufferString)

#add string cleanup here



#get lengths of all strings
stringLength = [0,0,0,0]
stringCount = 0
count = 0

#get string lengths for input strings (all that are currently supported)
while count < supportedStrings:
    stringLength[count] = len(stringList[count])
    count = count + 1

#check how many strings contain chars
stringCount = 0
count = 0
stringLengthCopy = stringLength
deleteStringList = []

while count < len(stringList):
    if stringLength[count]== 0:
        deleteStringList.append(count)
    count = count + 1

delStringLen = len(deleteStringList)
count = 1
while count < delStringLen:

    deleteStringList[count] = deleteStringList[count] - 1

    count = count + 1

count = 0
while count < delStringLen:
```

```
        del stringLength[deleteStringList[count]]
        del stringList[deleteStringList[count]]
        count = count + 1

    print stringLength
    #stick strings together with bufferString "glue"
    count = 0
    while count < len(stringList):

        if stringLength[count] == 0:
            count = count + 1
        else:
            foldString = foldString + stringList[count] + bufferString
            count = count + 1
        print foldString
    #take the last bufferString off the end of the foldString
    count = bufferStringLen


    print foldString


    foldString = foldString[0:len(foldString)-3]
    print foldString

    #create folding command with the --prohibit rule file
    #(command is actually -c=file)
    constraintString = ''



    ruleCount = stringCount - 1
    count = 0

    while count < ruleCount:
        incount = 0
        while incount < bufferStringLen:

            constraintString = constraintString + "P " + \
str(stringLength[ruleCount] + incount + 1) + "\r\n"

            incount = incount + 1

        count = count + 1


    #write constraints file
    cFile = open(constraintFile, 'w')

    cFile.write(constraintString)

    cFile.close()
```

```
    #create temp file
    rnaFile = open(rnaTempFile, 'w')

    rnaFile.write(foldString)

    rnaFile.close()

    #fold the structure
    commandString = "hybrid-ss-min --constraints=" + constraintFile + \
 " " + rnaTempFile
    os.system(commandString)
    #cleanup unnecessarily created files


    #rename and move desired file
#rename connection table for ease of graphing

    commandString = 'mv rnaTriQuadTemp.ct temp34.ct'
    os.system(commandString)

    #graph the fold

    commandString = 'sir_graph -p temp34.ct'
    os.system(commandString)

    #rename/move the fold visualization

    commandString = 'mv temp34.ps ' + str(g) + "/" + str(h)
    #print commandString
    os.system(commandString)


    #clean up some of the old files

    commandString = 'rm rnaTriQuadTemp*'
    os.system(commandString)

    commandString = 'rm temp34.ct'
    os.system(commandString)

    #display visualization

    commandString = 'evince --preview ' + str(g) + "/" + str(h) + " &"
    os.system(commandString)


    return
```

6 Author's Note

I apologise if anything in the preceding document is horribly wrong. Please do not take anything here as true before looking into things a bit for yourself. Also, because it seems to be the thing to do: if the code provided in the appendix wrecks your computer, you should realise that it was provided 'As is' and nobody promised you

that it would do that. I honestly can say I would be extremely surprised if it did, however. Also, as with all things, you should be reviewing the software licenses for all the programs that I've mentioned, just in case I misread/misunderstood something along the way. Good luck and happy folding!