*https://dx.doi.org/10.22161/eec.3.6.1* *ISSN: 2456-2319*

# A Program for Multiple Sequence Alignment by Star Alignment

Dibyaranjan Samal*[1], Nibedita Sahoo[1], Meesala Krishna Murthy[2]

[1]Department of Biotechnology, AMIT College, Khurda 752057, Odisha, India
[2]Department of Zoology, Mizoram University, Aizawl 796004, Mizoram, India
*Dibyaranjan Samal, Email: Dibyaranjan.daredivya@gmail.com

***Abstract*—** *Sequence alignment is the process of lining up two or more sequences to achieve maximal label of identity, further purpose of accessing is degree of similarity and the possibility of homology. Comparison of more than two string is known as multiple sequence alignment. Multiple sequence alignment (MSA) is a powerful tool in locating similar patterns in biological (DNA and Protein) sequences. The objective of this application is to develop a Multiple sequence alignment tool using star alignment method. This application / project has been developed using Visual Studio.Net and C# as page language was found to be appropriate for a rich user experience, faster code construction, automatic memory allocation and faster debugging.*

***Keywords*— *Star alignment, Multiple Sequence Alignment, Algorithms.***

## I. INTRODUCTION

The multiple sequence alignment is one of the challenging tasks in bioinformatics. It plays an essential role in finding conserved region among a set of sequences and inducing the evolutionary history and common properties of some species (Richer et al., 2007). It is known to be NP-complete and the current implementations of multiple alignment algorithms are heuristics owing to the high complexity on space and time needed when performing alignments (Elias & Isaac, 2006).Most existing algorithms for multiple sequence alignment are classified into three categories (Brudno et al., 2003). The first class is those algorithms that use high quality heuristics very close to optimality. They can only handle a small number of sequences with length less than 20 and limited to the sum-of-pairs objective function (Carrillo & Lipman, 1988). The second class is those algorithms using progressive alignments strategy and is by far the most widely used heuristics to align a large number of sequences (Sze et al., 2006). The third class is those algorithms using iterative refinement strategies to improve an initial alignment until convergence or reaching the maximum user-defined number of iterations (Hirosawa et al., 1995).We need heuristics to compute the MSA (Notredame, 2007). Usually, a heuristic does not guarantee the quality of the resulting alignment, it is faster, and in many cases, gives reasonably good answers. In star-alignment we build multiple alignment based on pairwise alignments between one of the sequences (call it the center of the star) and all the other sequences (Kleinjung, 2002).

This application has been developed in visual studio.net and C# as page language. XHTML, CSS is also used for better result display and adobe photoshop for animation (FIG 1).

## II. PROGRAM

Three Important program files have been used to get the result of Multiple Sequence alignment.

- Program.cs
- Input.cs
- Result.cs

**Program.cs:**

The **program.cs** file has been used for global configuration as to which file would run first.

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
namespace STARALIGNMENT
{
static class Program
  {
///<summary>
/// The main entry point for the application.
///</summary>
    [STAThread]
static void Main()
    {
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
```

*https://dx.doi.org/10.22161/eec.3.6.1* *ISSN: 2456-2319*

```
Application.Run(newresult());
      }
   }
}
```

**Input.cs:**

This is the input page where sequence is given to the tool. This is the important portion of this application as to dynamic programming algorithm is used in this form.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
namespace STARALIGNMENT
{
publicpartialclassINPUT : Form
   {
public INPUT()
      {
         InitializeComponent();
      }

privatevoid button2_Click(object sender, EventArgs e)
      {
this.Close();
      }


//GET MAX VALUE

privateint ReturnMax(int i, int j, int k)
      {
int retval = i;
if (j > retval)
         {
            retval = j;
         }
if (k > retval)
         {
            retval = k;
         }
return retval;
      }
//CONSTRUCT MATRIX
privatestring[,] ReturnArray(string seq)
      {
int gap = Convert.ToInt32(textBox7.Text);
int match = Convert.ToInt32(textBox5.Text);
int misMatch = Convert.ToInt32(textBox6.Text);

int cols = textBox1.Text.Length + 2;
int rows = seq.Length + 2;

string[,] ar = newstring[rows, cols];

for (int i = 0; i <= rows - 1; i++)
         {
for (int j = 0; j <= cols - 1; j++)
            {

if (i == 0 && (j == 0 || j == 1))
               {
                  ar[i, j] = " ";
               }
elseif (i == 0 && (j != 0 || j != 1))
               {
                  ar[i, j] = textBox1.Text[j -
2].ToString().ToUpper();
               }
elseif (j == 0 && i == 1)
               {
                  ar[i, j] = " ";
               }
elseif (j == 0 && i != 1)
               {
                  ar[i, j] = textBox2.Text[i -
2].ToString().ToUpper();
               }
elseif (j == 1 && i == 1)
               {
                  ar[i, j] = "0";
               }
elseif (i == 1 && j != 1)
               {
                  ar[i, j] =
Convert.ToString(Convert.ToInt32(ar[i, j - 1]) + gap);
               }
elseif (i != 1 && j == 1)
               {
                  ar[i, j] =
Convert.ToString(Convert.ToInt32(ar[i - 1, j]) + gap);
               }
else
               {
int val1 = Convert.ToInt32(ar[i - 1, j]) + gap;
```

*https://dx.doi.org/10.22161/eec.3.6.1* *ISSN: 2456-2319*

```
int val2 = Convert.ToInt32(ar[i, j - 1]) + gap;
int val3 = 0;
if (ar[0, j] == ar[i, 0])
                    {
                       val3 = Convert.ToInt32(ar[i - 1, j - 1]) +
match;
                    }
else
                    {
                       val3 = Convert.ToInt32(ar[i - 1, j - 1]) +
misMatch;
                    }
                    ar[i, j] = ReturnMax(val1, val2,
val3).ToString();
                 }
              }
           }
return ar;
      }
privatevoid button1_Click(object sender, EventArgs e)
      {

try
         {

result rr = newresult();
DataTable dt = newDataTable();
              dt.Columns.Add("SEQ",typeof(string));
              dt.Columns.Add("SCORE",typeof(int));

TextBox txtMatch = newTextBox();
              txtMatch =
(TextBox)rr.Controls["panel1"].Controls["textBox5"];
              txtMatch.Text = textBox5.Text;

TextBox txtMMatch = newTextBox();
              txtMMatch =
(TextBox)rr.Controls["panel1"].Controls["textBox6"];
              txtMMatch.Text = textBox6.Text;

TextBox txtG = newTextBox();
              txtG =
(TextBox)rr.Controls["panel1"].Controls["textBox7"];
              txtG.Text = textBox7.Text;

int gap = Convert.ToInt32(textBox7.Text);
int match = Convert.ToInt32(textBox5.Text);
int misMatch = Convert.ToInt32(textBox6.Text);
int cols = textBox1.Text.Length + 2;
int rows = textBox2.Text.Length + 2;
int rows1 = textBox3.Text.Length + 2;
int rows2 = textBox4.Text.Length + 2;

string[,] ar = newstring[rows, cols];
              ar = ReturnArray(textBox2.Text);

string[,] ar1 = newstring[rows1, cols];
              ar1 = ReturnArray(textBox3.Text);

string[,] ar2 = newstring[rows2, cols];
              ar2 = ReturnArray(textBox4.Text);

FileStream fs = newFileStream(Application.StartupPath +
"/rpt.html", FileMode.Create);
StreamWriter sw = newStreamWriter(fs);

              sw.WriteLine("<table border='1' bordercolor='red'
width='100%'>");
for (int i = 0; i <= rows - 1; i++)
              {
                 sw.WriteLine("<tr>");
for (int j = 0; j <= cols - 1; j++)
                 {
                    sw.WriteLine("<td>");
                    sw.WriteLine(ar[i, j]);
                    sw.WriteLine("</td>");
                 }
                 sw.WriteLine("</tr>");
              }
              sw.WriteLine("</table>");
              sw.WriteLine("<br />");
              sw.WriteLine("SEQUENCE ALLIGNMENT:");
string seq = GetAlignment(ar, rows, cols);
string[] aseq = seq.Split('^');
int finalScore = 0;

              sw.WriteLine("<table>");
              sw.WriteLine("<tr>");
for (int i = aseq.Length - 1; i >= 0; i--)
              {
                 sw.WriteLine("<td>");
                 sw.WriteLine(aseq[i].ToString()[0] + "<br />"
+ aseq[i].ToString()[1]);
if (aseq[i].ToString()[0] == aseq[i].ToString()[1])
                 {
                    finalScore = finalScore + match;
                 }
```

*https://dx.doi.org/10.22161/eec.3.6.1* *ISSN: 2456-2319*

```
if (aseq[i].ToString()[0]  != aseq[i].ToString()[1])
                {
                   finalScore  = finalScore  + misMatch;
                }
if (aseq[i].ToString()[0].ToString()  == "_" ||
aseq[i].ToString()[1].ToString()  == "_")
                {
                   finalScore  = finalScore  + gap;
                }
                sw.WriteLine("</td>");
             }
             sw.WriteLine("</tr>");
             sw.WriteLine("</table>");
             sw.WriteLine("<br  />");
             sw.WriteLine("Score:  " + finalScore);
             sw.WriteLine("<br  /><br />");
             dt.Rows.Add("SEQ1",finalScore);
             sw.WriteLine("<table  border='1' bordercolor='red'
width='100%'>");
for (int i = 0; i <= rows1 - 1; i++)
             {
                sw.WriteLine("<tr>");
for (int j = 0; j <= cols - 1; j++)
                {
                   sw.WriteLine("<td>");
                   sw.WriteLine(ar1[i,  j]);
                   sw.WriteLine("</td>");
                }
                sw.WriteLine("</tr>");
             }
             sw.WriteLine("</table>");
             sw.WriteLine("<br  />");
             sw.WriteLine("<br  />");
             sw.WriteLine("SEQUENCE  ALLIGNMENT:");
string seq1 = GetAlignment(ar1,  rows1, cols);
string[] aseq1 = seq1.Split('^');
int finalScore1  = 0;


             sw.WriteLine("<table>");
             sw.WriteLine("<tr>");
for (int i = aseq1.Length - 1; i >= 0; i--)
             {
                sw.WriteLine("<td>");
                sw.WriteLine(aseq1[i].ToString()[0]  + "<br />"
+ aseq1[i].ToString()[1]);
if (aseq1[i].ToString()[0]  == aseq1[i].ToString()[1])
                {
                   finalScore1  = finalScore1  + match;
                }
```

```
if (aseq1[i].ToString()[0]  != aseq1[i].ToString()[1])
                {
                   finalScore1  = finalScore1  + misMatch;
                }
if (aseq1[i].ToString()[0].ToString()  == "_" ||
aseq1[i].ToString()[1].ToString()  == "_")
                {
                   finalScore1  = finalScore1  + gap;
                }
                sw.WriteLine("</td>");
             }
             sw.WriteLine("</tr>");
             sw.WriteLine("</table>");
             sw.WriteLine("<br  />");
             sw.WriteLine("Score:  " + finalScore1);
             sw.WriteLine("<br  /><br />");
             dt.Rows.Add("SEQ2",  finalScore1);
             sw.WriteLine("<table  border='1' bordercolor='red'
width='100%'>");
for (int i = 0; i <= rows2 - 1; i++)
             {
                sw.WriteLine("<tr>");
for (int j = 0; j <= cols - 1; j++)
                {
                   sw.WriteLine("<td>");
                   sw.WriteLine(ar2[i,  j]);
                   sw.WriteLine("</td>");
                }
                sw.WriteLine("</tr>");
             }
             sw.WriteLine("</table>");
             sw.WriteLine("<br  />");
             sw.WriteLine("<br  />");
             sw.WriteLine("SEQUENCE  ALLIGNMENT:");
string seq2 = GetAlignment(ar2,  rows2, cols);
string[] aseq2 = seq2.Split('^');
int finalScore2  = 0;
             sw.WriteLine("<table>");
             sw.WriteLine("<tr>");
for (int i = aseq2.Length - 1; i >= 0; i--)
             {
                sw.WriteLine("<td>");
                sw.WriteLine(aseq2[i].ToString()[0]  + "<br />"
+ aseq2[i].ToString()[1]);
if (aseq2[i].ToString()[0]  == aseq2[i].ToString()[1])
                {
                   finalScore2  = finalScore2  + match;
                }
if (aseq2[i].ToString()[0]  != aseq2[i].ToString()[1])
```

*https://dx.doi.org/10.22161/eec.3.6.1* *ISSN: 2456-2319*

```
                {
                    finalScore2 = finalScore2 + misMatch;
                }
if (aseq2[i].ToString()[0].ToString() == "_" ||
aseq2[i].ToString()[1].ToString() == "_")
                {
                    finalScore2 = finalScore2 + gap;
                }
                sw.WriteLine("</td>");
            }
            sw.WriteLine("</tr>");
            sw.WriteLine("</table>");
            sw.WriteLine("<br />");
            sw.WriteLine("Score: " + finalScore2);
            sw.WriteLine("<br /><br />");
            dt.Rows.Add("SEQ3", finalScore2);
            sw.WriteLine("<table cellpadding='0'
cellspacin='0'>");
            sw.WriteLine("<tr>");
            sw.WriteLine("<td>Central Seq</td>");
            sw.WriteLine("</tr>");
//PHYLOGENY
DataRow[] filtered = dt.Select("", "SCORE DESC",
DataViewRowState.Added);
for (int s = 0; s <= dt.Rows.Count - 1; s++)
            {
                sw.WriteLine("<tr>");
                sw.WriteLine("<td style='text-aling:center;'>");
                sw.WriteLine("|<br />");
                sw.WriteLine("|");
                sw.WriteLine("</td>");
                sw.WriteLine("</tr>");

                sw.WriteLine("<tr>");

sw.WriteLine("<td>"+dt.Rows[s]["SEQ"]+"</td>");
                sw.WriteLine("</tr>");
            }
            sw.WriteLine("</table>");
            sw.Close();
            fs.Close();
            rr.Show();
this.Hide();

        }
catch(Exception ex)
        {
MessageBox.Show(ex.Message);
        }
    }

//TRACE BACK & GET ALIGNMENT

privatestring GetAlignment(string [,]ar,int rows,int cols)
    {
int gap = Convert.ToInt32(textBox7.Text);
int match = Convert.ToInt32(textBox5.Text);
int misMatch = Convert.ToInt32(textBox6.Text);
string seq = string.Empty;
int ii = rows - 1;
int jj = cols - 1;

//TRACEBACK
while (ii != 1 || jj != 1)
        {
int val1 = Convert.ToInt32(ar[ii - 1, jj]) + gap;
int val2 = Convert.ToInt32(ar[ii, jj - 1]) + gap;
int val3 = 0;
if (ar[0, jj] == ar[ii, 0])
            {
                val3 = Convert.ToInt32(ar[ii - 1, jj - 1]) +
match;
            }
else
            {
                val3 = Convert.ToInt32(ar[ii - 1, jj - 1]) +
misMatch;
            }
            ar[ii, jj] = ReturnMax(val1, val2,
val3).ToString();
if (ar[ii, jj] == val1.ToString())
            {
                seq += "_" + ar[ii, 0] + "^";
                ii = ii - 1;
            }
elseif (ar[ii, jj] == val2.ToString())
            {
                seq += ar[0, jj] + "_" + "^";
                jj = jj - 1;
            }
elseif (ar[ii, jj] == val3.ToString())
            {
                seq += ar[0, jj] + ar[ii, 0] + "^";
                ii = ii - 1;
                jj = jj - 1;
            }
        }
```

*https://dx.doi.org/10.22161/eec.3.6.1* *ISSN: 2456-2319*

```
            seq = seq.Substring(0, seq.Length - 1);
return seq;
        }
    }
}
```

## III. RESULTS

This is the result form of this tool where a XHTML document is written using filehandling (SYSTEM.IO namespace) and displayed on this application.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace STARALIGNMENT
{
publicpartialclassresult : Form
    {
public result()
        {
            InitializeComponent();
        }
privatevoid button2_Click_1(object sender, EventArgs e)
        {
this.Close();
        }
privatevoid result_Load(object sender, EventArgs e)
        {
            webBrowser1.Navigate(Application.StartupPath +
"/rpt.html");
        }
    }
}
```

## IV. CONCLUSION AND FUTURE SCOPE

This program only convert the star alignment method to program where fixed number of sequence are taken into consideration and we are also assuming our own central sequence for programming efficiency of the method. This should be done dynamically, where the central sequence should be selected and as many multiple numbers of sequences has to be taken. A single database of protein, DNA and RNA repository has to be made in the next version of this Project / Application.

## REFERENCES

[1] Richer J.M. Derrien V., Hao J.K. (2007). A New Dynamic Programming Algorithm for Multiple Sequence Alignment. COCOA LNCS, 4616, 52-61.

[2] Elias, Isaac. (2006). Settling The Intractability Of Multiple Alignment. Journal of Computational Biology, **13(7),** 1323-1339.

[3] Brudno M.,Chapman M., Gottgens B., Batzoglou S., Morgenstern B. (2003). Fast And Sensitive Multiple Alignments Of Large Genomic Sequences. BMC Bioinformatics, **4,** 66.

[4] Carrillo H., Lipman D.J. (1988). The Multiple Sequence Alignment Proelem in Biology. SIAM Journal of Applied Mathematics, **48(5),** 1073-1082.

[5] Sze S.H., Lu Y., Yang Q. A. (2006). Polynomial Tie Solvable Formulation of Multiple Sequence Aignment. Journal of Computational Biology, **13(2),** 309-319.

[6] Hirosawa M., Totoki Y., Hoshida M., Ishikawa M. (1995). Comprehensive Study On Iterative Algorithms Of Multiple Sequence Alignment. Comput Appl Biosci, **11,** 13-18.

[7] Notredame C. (2007). Recent Evolutions of Multiple Sequence Alignment Algoritms.PLOS Computational Biology, **3(8:e123),** 1405-1408.

[8] Kleinjung J., DouglasN., Heringa J. (2002). Multiple Sequence Alignments with parallel Computing. Bioinformatics, **18(9),** 1270-1271.

*https://dx.doi.org/10.22161/eec.3.6.1* *ISSN: 2456-2319*

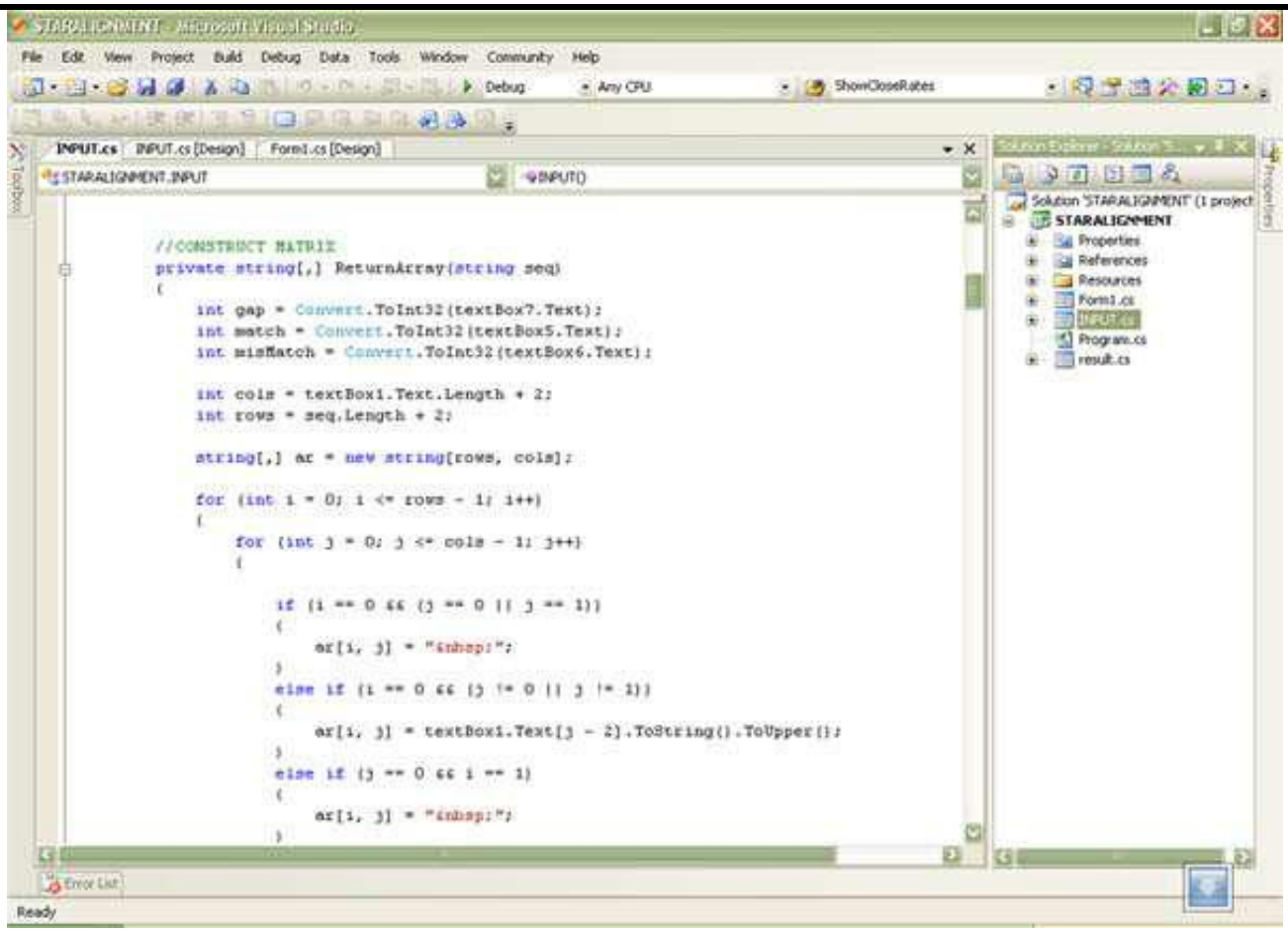


*Fig.1: The Visual studio .net Integrated Development Environment where the application has been developed.*