SIMULATING THE RECONSTRUCTION OF METABOLIC NETWORKS USING MAPLE

DIEGO IGNACIO VELEZ JARAMILLO
COMPUTATIONAL AND THEORICAL PHYSICS GROUP
LOGIG AND COMPUTACIONAL GROUP
ENGENERING PHYSCIS PROGRAM
EAFIT UNIVERSITY
MEDELLIN, COLOMBIA

Abstract *- In this article is simulated the reconstruction of metabolic networks by means of an algorithm in maple that represents the genome and its form of expression. The metabolic networks are represented by means of graphs, which are constructed from their matrices of adjacency (first squared that represents the connections between elements). Using the adjacency matrix also constructs a graph of density by means in which the algorithm for the reconstruction of the gene is applied.*

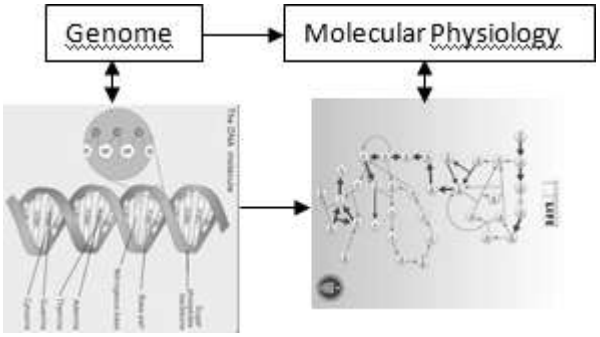
Keywords: Genome, graphs, metabolic networks, molecular physiology, maple.

1 Introduction

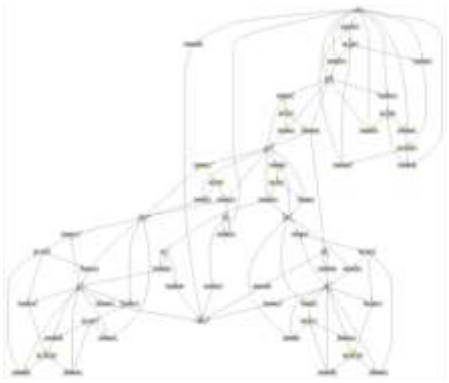
A gene is an organized linear sequence of nucleotides that contains the necessary information for the synthesis of a macro-molecule with specific cellular function, normally proteins. The proteins occupy a place of maximum importance between constituent molecules of the alive beings, practically all the biological processes depend on the presence or the activity of this type of molecule. Due to this the importance of studying the form in which the genes manipulate this information and the form that the macromolecules express themselves. Also the expression form can be manipulated to take macro-molecules to a wished protein.

2 Problem

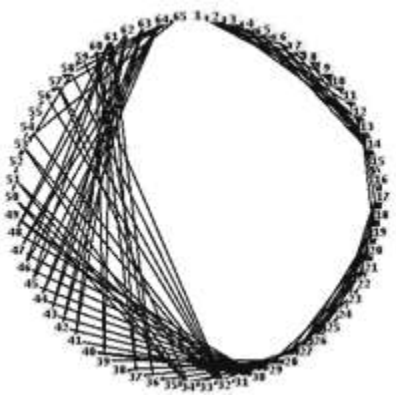
Show how from the genomes is reconstructed molecular physiology [1,2].



2.1 Molecular Physiology treat



$g := Graph(\{\{1,2\},\{1,3\},\{1,4\},\{1,5\},\{1,6\},\{1,7\},\{1,8\},\{1,9\},\{1,10\},\{1,11\},\{2,12\},\{2,13\},\{3,13\},\{3,14\},\{4,14\},\{4,15\},\{5,14\},\{5,15\},\{6,12\},\{6,15\},\{7,12\},\{7,15\},\{8,12\},\{8,13\},\{9,14\},\{9,13\},\{10,16\},\{11,17\},\{12,18\},\{12,19\},\{12,20\},\{16,21\},\{16,22\},\{16,23\},\{16,24\},\{16,25\},\{17,23\},\{17,11\},\{17,26\},\{18,27\},\{18,28\},\{19,27\},\{19,28\},\{19,29\},\{20,28\},\{20,27\},\{21,30\},\{22,29\},\{24,31\},\{25,32\},\{26,28\},\{26,33\},\{26,34\},\{28,35\},\{28,36\},\{28,37\},\{28,38\},\{28,39\},\{29,40\},\{40,29\},\{30,40\},\{30,41\},\{30,42\},\{30,43\},\{30,44\},\{30,45\},\{30,46\},\{30,47\},\{30,48\},\{31,34\},\{31,49\},\{49,31\},\{32,49\},\{32,50\},\{32,51\},\{32,59\},\{32,53\},\{32,54\},\{32,57\},\{32,58\},\{32,56\},\{33,35\},\{33,38\},\{34,35\},\{34,38\},\{34,53\},\{34,57\},\{34,50\},\{34,51\},\{36,60\},\{36,61\},\{37,60\},\{37,61\},\{39,60\},\{39,61\},\{41,61\},\{41,62\},\{42,62\},\{42,63\},\{43,62\},\{43,63\},\{44,63\},\{44,64\},\{45,61\},\{45,64\},\{46,61\},\{46,64\},\{47,63\},\{47,64\},\{48,61\},\{48,62\},\{50,52\},\{51,52\},\{52,54\},\{52,59\},\{53,65\},\{54,55\},\{55,56\},\{55,59\},\{55,58\},\{56,65\},\{58,65\},\{57,65\}\});$



3 Method

1. Represent metabolic networks as graphs, defined by its adjacency matrices
2. Represent the adjacency matrix as a graph of density
3. Represent the genome using the following algorithm for image reconstruction.

The following algorithm represents the genome and how it manipulates information to reconstruct the molecular physiology.

```
List_Density_Plot := proc(data)
  global i_row,j_column;
    local i,f, local_Am;
    f := x→-x + 1;
     local_Am := array(1 ..i_row, 1 ..j_column, [seq(map(f,
      [seq(data[i_row-i + 1,j],j = 1 ..j_column)]), i = 1 ..i_row)]) :
     listdensityplot(transpose(local_Am));
   end:

ns_sum := proc( )
    global i_row,j_column, Am;
    local i,j, ns;
    for j from 1 to j_column do
      ns[j] := 0 :
      for i from 1 to i_row do
        ns[j] := ns[j] + Am[i,j];
      od;
    od;
    RETURN(ns);
  end:

we_sum := proc( )
     global i_row,j_column, Am;
     local i,j, we;
     for i from 1 to i_row do
       we[i] := 0 :
       for j from 1 to j_column do
         we[i] := we[i] + Am[i,j];
       od;
     od;
     RETURN(we);
   end:

NwSe_diag_sum := proc( )
    global i_row,j_column, Am;
     local i,j, we_diag;
     for i from 1 to i_row do
       we_diag[i, 1] := 0 : we_diag[i, 2] := 0 :
       for j from 1 to 100 while ((j ≤j_column) and (i-j + 1 ≥ 1)) do
          we_diag[i, 1] := we_diag[i, 1] + Am[i-j + 1,j];
         we_diag[i, 2] := we_diag[i, 2] + 1;
       od;
     od;
     for j from 2 to j_column do
       we_diag[i_row + j-1, 1] := 0 : we_diag[i_row + j-1, 2] := 0 :
      for i from 1 to 100 while ((j + i-1 ≤j_column) and (i_row-i
       + 1 ≥ 1)) do
          we_diag[i_row + j-1, 1] := we_diag[i_row + j-1, 1]
       + Am[i_row-i + 1,j + i-1];
          we_diag[i_row + j-1, 2] := we_diag[i_row + j-1, 2] + 1;
       od;
     od:
     RETURN(we_diag);
   end:

NeSw_diag_sum := proc( )
     global i_row,j_column, Am;
     local i,j, ew_diag;
     for i from 1 to i_row do
       ew_diag[i, 1] := 0 : ew_diag[i, 2] := 0 :
       for j from 1 to 100 while ((j_column-j + 1 ≥ 1) and (i-j + 1
       ≥ 1)) do
          ew_diag[i, 1] := ew_diag[i, 1] + Am[i-j + 1,j_column-j
       + 1];
          ew_diag[i, 2] := ew_diag[i, 2] + 1;
       od;
     od;
     for j from 2 to j_column do
       ew_diag[i_row + j-1, 1] := 0 : ew_diag[i_row + j-1, 2] := 0 :
       for i from 1 to 100 while ((j_column-i-j + 2 ≥ 1) and (i_row-i
       + 1 ≥ 1)) do
          ew_diag[i_row + j-1, 1] := ew_diag[i_row + j-1, 1]
       + Am[i_row-i + 1,j_column-i-j + 2];
          ew_diag[i_row + j-1, 2] := ew_diag[i_row + j-1, 2] + 1;
       od;
     od:
    RETURN(ew_diag);
   end:

vertical_distrib := proc(a)
      global i_row,j_column;
      local i,j, temp_array, temp_matrix;
      for j from 1 to j_column do
        for i from 1 to i_row do
          temp_array[i,j] := a[j]/i_row;
        od;
      od;
      temp_matrix := array(1 ..i_row, 1 ..j_column,
      [seq([seq(temp_array[i,j],j = 1 ..j_column)], i = 1 ..i_row)]);
      RETURN(temp_matrix);
   end:

horizontal_distrib := proc(a)
     global i_row,j_column;
     local i,j, temp_array, temp_matrix;
     for i from 1 to i_row do
       for j from 1 to j_column do
          temp_array[i,j] := a[i]/j_column;
       od;
     od;
     temp_matrix := array(1 ..i_row, 1 ..j_column,
     [seq([seq(temp_array[i,j],j = 1 ..j_column)], i = 1 ..i_row)]);
     RETURN(temp_matrix);
   end:
```

```
NwSe_diag_distrib := proc(a)
  global i_row, j_column;
  local i, j, temp_array, temp_matrix;
  for i from 1 to i_row do
    for j from 1 to 100 while ((j ≤ j_column) and (i-j + 1 ≥ 1)) do
      temp_array[i-j + 1, j] := a[i, 1]/a[i, 2];
    od;
  od;
  for j from 2 to j_column do
    for i from 1 to 100 while ((j + i-1 ≤ j_column) and (i_row-i
    + 1 ≥ 1)) do
      temp_array[i_row-i + 1, j + i-1] := a[i_row + j-1, 1]
   /a[i_row + j-1, 2];
    od;
  od:
  temp_matrix := array(1 ..i_row, 1 ..j_column,
   [seq([seq(temp_array[i, j], j = 1 ..j_column)], i = 1 ..i_row)]);
  RETURN(temp_matrix);
end:
```

```
NeSw_diag_distrib := proc(a)
  global i_row, j_column;
  local i, j, temp_array, temp_matrix;
  for i from 1 to i_row do
    for j from 1 to 100 while ((j_column-j + 1 ≥ 1) and (i-j + 1
   ≥ 1)) do
      temp_array[i-j + 1, j_column-j + 1] := a[i, 1]/a[i, 2];
    od;
  od;
  for j from 2 to j_column do
    for i from 1 to 100 while ((j_column-i-j + 2 ≥ 1) and (i_row-i
    + 1 ≥ 1)) do
      temp_array[i_row-i + 1, j_column-i-j + 2] := a[i_row + j
   -1, 1]/a[i_row + j-1, 2];
    od;
  od:
  temp_matrix := array(1 ..i_row, 1 ..j_column,
   [seq([seq(temp_array[i, j], j = 1 ..j_column)], i = 1 ..i_row)]);
  RETURN(temp_matrix);
end:
```

4 Results

$$\text{>G := SoccerBallGraph( );}$$

Graph 1: an undirected unweighted graph with 60 vertices and 90 edge(s)

$$i\_row := 60 : j\_column := 60 :$$
$$catscan := AdjacencyMatrix(G)$$
$$\begin{bmatrix} 60\ x\ 60\ Matrix \\ Data\ Type:\ anything \\ Storage:\ triangular_{upper} \\ Order:\ C\_order \end{bmatrix}$$

$$we := we\_sum( ) : ns := ns\_sum( ) : NwSe\_diag := NwSe\_diag\_sum( ) : NeSw\_diag := NeSw\_diag\_sum( ) :$$

$$seq(we[i], i = 1 ..i\_row) : seq(ns[j], j = 1 ..j\_column) : seq(NwSe\_diag[i, 1], i = 1 ..i\_row + j\_column-1) : seq(NeSw\_diag[i, 1], i = 1 ..i\_row + j\_column-1) :$$

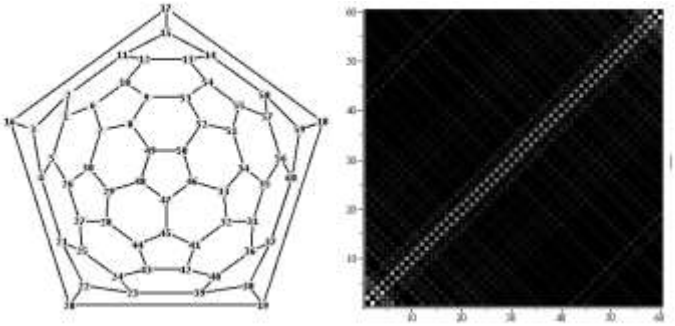
$$seq(NwSe\_diag[i, 2], i = 1 ..i\_row + j\_column-1) : seq(NeSw\_diag[i, 2], i = 1 ..i\_row + j\_column-1) :$$

$$img[1] := vertical\_distrib(ns) : img[2] := horizontal\_distrib(we) : img[3] := NwSe\_diag\_distrib(NwSe\_diag) : img[4] := NeSw\_diag\_distrib(NeSw\_diag) :$$

$$\#\ print([seq(ns[j],j=1..j\_column)]): print(img[1]): total\_image := (evalm(sum(img[i], i = 1 ..4))) :$$

$$threshold := sort([seq(seq(total\_image[i, j], j = 1 ..j\_column), i = 1 ..i\_row)], `>`)[sum(we[j], j = 1 ..i\_row)] :$$

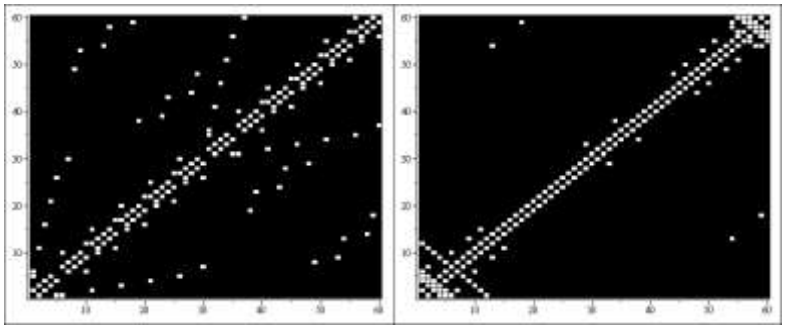
$$\text{> listdensityplot(total\_image);}$$



> $g := x \rightarrow \textbf{if}\ (x \geq threshold)\ \textbf{then}\ 1\ \textbf{else}\ 0\ \textbf{fi}$:

$p1 := listdensityplot(catscan) : p2 := listdensityplot(map(g, total\_image)) :$

> $display(array([p1, p2]))$;



> $H := FosterGraph(\,)$;

Graph 4: an undirected unweighted graph with 90 vertices and 135 edge(s)

> $i\_row := 90 : j\_column := 90 :$
$Am := AdjacencyMatrix(H) :$

$we := we\_sum(\,) : ns := ns\_sum(\,) : NwSe\_diag := NwSe\_diag\_sum(\,) : NeSw\_diag := NeSw\_diag\_sum(\,) :$

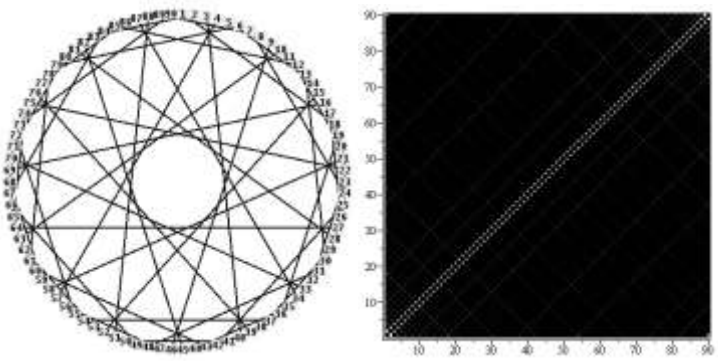
$seq(we[i], i = 1 .. i\_row) : seq(ns[j], j = 1 .. j\_column) :$
$seq(NwSe\_diag[i, 1], i = 1 .. i\_row + j\_column - 1) :$
$seq(NeSw\_diag[i, 1], i = 1 .. i\_row + j\_column - 1) :$

$seq(NwSe\_diag[i, 2], i = 1 .. i\_row + j\_column - 1) : seq(NeSw\_diag[i, 2], i = 1 .. i\_row + j\_column - 1) :$

$img[1] := vertical\_distrib(ns) : img[2] := horizontal\_distrib(we) :$
$img[3] := NwSe\_diag\_distrib(NwSe\_diag) : img[4] := NeSw\_diag\_distrib(NeSw\_diag) :$

$\#\ print([seq(ns[j],j=1..j\_column)]): print(img[1]):$
$total\_image := (evalm(sum(img[i], i = 1 .. 4))) :$
$threshold := sort([seq(seq(total\_image[i, j], j = 1 .. j\_column), i = 1 .. i\_row)], `>`)[sum(we[j], j = 1 .. i\_row)] :$

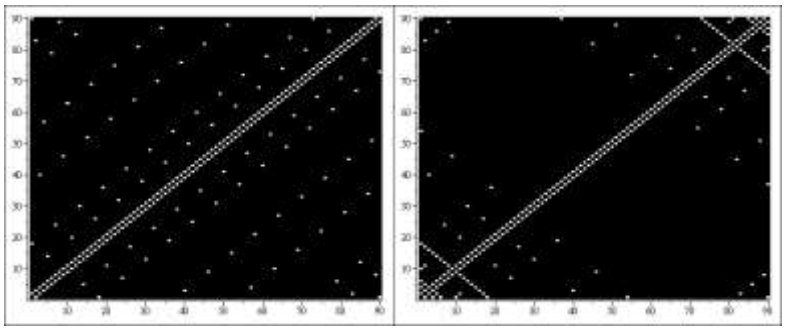
> $listdensityplot(total\_image); DrawGraph(H);$



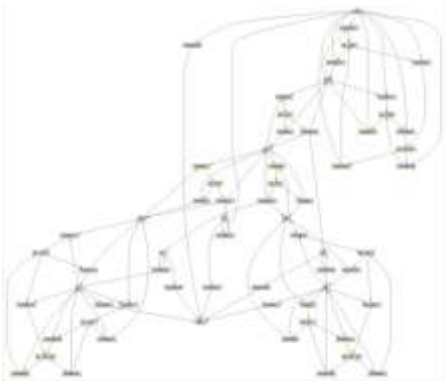
> $g := x \rightarrow \textbf{if } (x \geq threshold) \textbf{ then } 1 \textbf{ else } 0 \textbf{ fi}:$

$p1 := listdensityplot(catscan) : p2 := listdensityplot(map(g, total\_image)) :$

> $display(array([p1, p2]));$



Metabolic network Taken for analysis



$g := Graph(\{\{1,2\},\{1,3\},\{1,4\},\{1,5\},\{1,6\},\{1,7\},\{1,8\},\{1,9\},\{1,10\},\{1,11\},\{2,12\},\{2,13\},\{3,13\},\{3,14\},\{4,14\},\{4,15\},\{5,14\},\{5,15\},\{6,12\},\{6,15\},\{7,12\},\{7,15\},\{8,12\},\{8,13\},\{9,14\},\{9,13\},\{10,16\},\{11,17\},\{12,18\},\{12,19\},\{12,20\},\{16,21\},\{16,22\},\{16,23\},\{16,24\},\{16,25\},\{17,23\},\{17,11\},\{17,26\},\{18,27\},\{18,28\},\{19,27\},\{19,28\},\{19,29\},\{20,28\},\{20,27\},\{21,30\},\{22,29\},\{24,31\},\{25,32\},\{26,28\},\{26,33\},\{26,34\},\{28,35\},\{28,36\},\{28,37\},\{28,38\},\{28,39\},\{29,40\},\{40,29\},\{30,40\},\{30,41\},\{30,42\},\{30,43\},\{30,44\},\{30,45\},\{30,46\},\{30,47\},\{30,48\},\{31,34\},\{31,49\},\{49,31\},\{32,49\},\{32,50\},\{32,51\},\{32,59\},\{32,53\},\{32,54\},\{32,57\},\{32,58\},\{32,56\},\{33,35\},\{33,38\},\{34,35\},\{34,38\},\{34,53\},\{34,57\},\{34,50\},\{34,51\},\{36,60\},\{36,61\},\{37,60\},\{37,61\},\{39,60\},\{39,61\},\{41,61\},\{41,62\},\{42,62\},\{42,63\},\{43,62\},\{43,63\},\{44,63\},\{44,64\},\{45,61\},\{45,64\},\{46,61\},\{46,64\},\{47,63\},\{47,64\},\{48,61\},\{48,62\},\{50,52\},\{51,52\},\{52,54\},\{52,59\},\{53,65\},\{54,55\},\{55,56\},\{55,59\},\{55,58\},\{56,65\},\{58,65\},\{57,65\}\});$

> $i\_row := 65 : j\_column := 65 :$

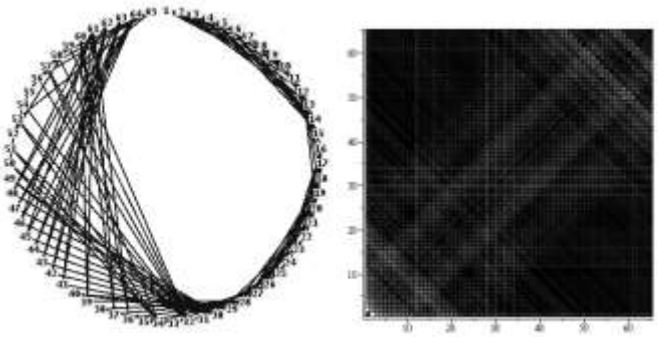
$Am := AdjacencyMatrix(g) :$

$we := we\_sum( ) : ns := ns\_sum( ) : NwSe\_diag := NwSe\_diag\_sum( ) : NeSw\_diag := NeSw\_diag\_sum( ) :$

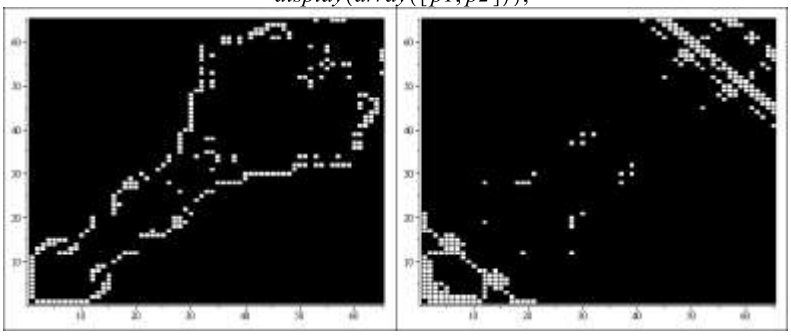
$seq(we[i], i = 1 .. i\_row) : seq(ns[j], j = 1 .. j\_column) :$
$seq(NwSe\_diag[i, 1], i = 1 .. i\_row + j\_column - 1) :$
$seq(NeSw\_diag[i, 1], i = 1 .. i\_row + j\_column - 1) :$

$seq(NwSe\_diag[i, 2], i = 1 .. i\_row + j\_column - 1) : seq(NeSw\_diag[i, 2], i = 1 .. i\_row + j\_column - 1) :$

```
img[1] := vertical_distrib(ns) : img[2] := horizontal_distrib(we) :
    img[3] := NwSe_diag_distrib(NwSe_diag) : img[4]
    := NeSw_diag_distrib(NeSw_diag) :

    # print([seq(ns[j],j=1..j_column)]): print(img[1]):
    total_image := (evalm(sum(img[i], i = 1 ..4))) :
threshold := sort([seq(seq(total_image[i, j], j = 1 ..j_column), i = 1
    ..i_row)], `>`)[sum(we[j], j = 1 ..i_row)] :

> listdensityplot(total_image); DrawGraph(g);
```



```
> g := x → if (x ≥ threshold) then 1 else 0 fi:
p1 := listdensityplot(Am) : p2 := listdensityplot(map(g,
    total_image)) :

> display(array([p1, p2]));
```



5 Conclusions

In this article I simulate the reconstruction of metabolic networks by means of an algorithm that represents the information codified within a gene. By means of this information it is possible to know the form that a macro-molecule is processed and representing it by means of a dens dad graph. In the development process many reconstructions were realized beginning by graphs of low complexity in which was observed that the realized reconstructions were exact, as the complexity of the interactions are increased and these behave in a asymmetric form, it's observed that the reconstructions move away a little from their original structure. These discrepancies are acceptable considering that conserve the central structures. This can be interpreted like the variation between phenotypes from an original genotype.

6 References

1. HTTP://WWW.MATHWORKS.COM/PRODUCTS/BIOINFO/DEMOS.HTML?FILE=/PRODUCTS/DEMOS/SHIPPING/BIOINFO/GRAPHTHEORYDEMO.HTML
2. HTTP://EN.WIKIPEDIA.ORG/WIKI/METABOLIC_NETWORK_MODELLING
3. HTTP://WWW.MAPLESOFT.COM/APPLICATIONS/VIEW.ASPX?SID=4273&VIEW=HTML
4. GRAPH THEORY MAPLE SOFTWARE.