MetFBA: A WEB BASED TOOL FOR FLUX BALANCE ANALYSIS

Project report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Bioinformatics

Under the supervision of

Dr. Ragothaman M. Yennamalli

Dr. Tiratha Raj Singh

Submitted by

Shubhangi Kaushik(131517)

Devesh Thapar(131507)

DEPARTMENT OF BIOTECHNOLOGY AND BIOINFORMATICS

JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY

WAKNAGHAT, HIMACHAL PRADESH 173234

CERTIFICATE

This is to certify that the project report entitled "MetFBA: A WEB BASED TOOL FOR FLUX BALANCE ANALYSIS**",** submitted by **DEVESH THAPAR** and **SHUBHANGI KAUSHIK** in partial fulfillment for the award of degree of Bachelor of Technology in Bioinformatics Engineering to Jaypee University of Information Technology, Waknaghat has been carried out under our supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date :

Dr. Ragothaman M. Yennamalli

Assistant Professor (Grade II)

Dr. Tiratha Raj Singh

Assistant Professor (Senior Grade)

ACKNOWLEDGEMENT

This work would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost, our utmost gratitude to **Dr. Ragothaman M. Yennamalli**, Assistant Professor (Grade - II) and **Dr. Tiratha Raj Singh**, Assistant Professor (Senior Grade) at the Department of Biotechnology and Bioinformatics (JUIT), for their unfailing support as our project advisers and for their patience and steadfast encouragement to carry on with this study.

Abbreviations

FBA – Flux Balance Analysis

SBRT – Systems Biology Research Tool

MATLAB – Matrix Laboratory

SDLC – Software Development Life Cycle

GUI – Graphical User Interface

HTML – Hypertext Markup Language

CSS – Cascading Style Sheets

XAMPP – (X stands for cross platform) Apache, MySQL, PHP, Perl

CGI – Common Gateway Interface

SBML – Systems Biology Markup Language

LPP – Linear Programming Problem

GLPK – GNU Linear Programming Kit

Abstract

Flux Balance Analysis (FBA) is a constraint based approach used for the analysis of metabolic networks. FBA calculates the flow of metabolites through the metabolic networks. This method enforces constraints on a metabolic system and using mathematical approach of linear programming, gives optimal solution for a given objective function. FBA has been gaining ground since the past few years because of increase in the availability of pathway data in multiple databases.

We have developed a web based tool, MetFBA (**Met**abolic **F**lux **B**alance **A**nalysis), for analyzing the metabolic networks. For the optimization, we have used GNU Linear Programming Kit (GLPK). This tool gives the final output as the optimized value of the objective function and also the values involved with each of the fluxes.

Contents

CHAPTER 1 - Introduction

1.1 Background:

With the large amount of data, that is available in various biochemical pathway databases and biochemical data stored in the literature, it is possible to apply various approaches for the analysis and simulation of metabolic pathways. Some of these approaches are: (a) interaction based methods, (b) mechanistic approach and (c) constraint based methods [1]. Interaction based methods uses a graphical approach to map metabolic networks, whereas mechanist approach uses stoichiometry and kinetic parameters. The drawback to these approaches is the lack of all kinetic parameters for all the reactions. The constraint based method is the most popular. The approach uses constraints like physico-chemical, gene regulatory and environmental constraints. Using the available data one can model the biochemical pathway as the first step. Once the pathway has been modeled we can apply any of the three approaches to study the various aspects of metabolic networks, metabolism and cellular operations. The analysis and simulation of these networks is used mainly for drug discovery, in food industry and in bio-engineering where the aim is to increase the functionality of micro-organisms [2].

In this project the approach we are focusing on is **Flux Balance Analysis,** a constraint-based approach to develop a web based tool with enhanced usability and user friendly implementation of intermediate processes to generate robust and correct output.

1.2 Constraint Based Approach

In general, a constraint based method enforces constraints on a metabolic system [3]. The success of this methodology is due to the availability of the resources required for reconstruction of effective metabolic networks [1]. Some of these resources are mentioned in following table:

Table 1 : Resources that are available for reconstruction of metabolic networks:

Resource	URL	Reference
Kyoto Encyclopedia of Genes and Genomes(KEGG)	http://www.genome.jp/kegg/	[4]
BioCyc	http://www.biocyc.org/	[5]
Reactome	http://reactome.org/	[6]
BRENDA	http://brenda-enxymes.info/	[7]

The genome-wide annotation of protein sequences is used as the foundation in metabolic network reconstruction. This is possible due to constraint-based method, Flux Balance Analysis, which requires only stoichiometry information for carrying out the analysis of biological networks. Thus, such approach requires very less information as compared to others.

1.3 Flux Balance Analysis

Flux Balance Analysis (FBA) generally is carried out by doing steady state analysis of a system using the stoichiometry matrix. The steady state analysis requires the creation of an objective function. The objective function is created by using the assumption that the cell performs optimally at steady state [8]. FBA consist of the following steps (Figure 1):

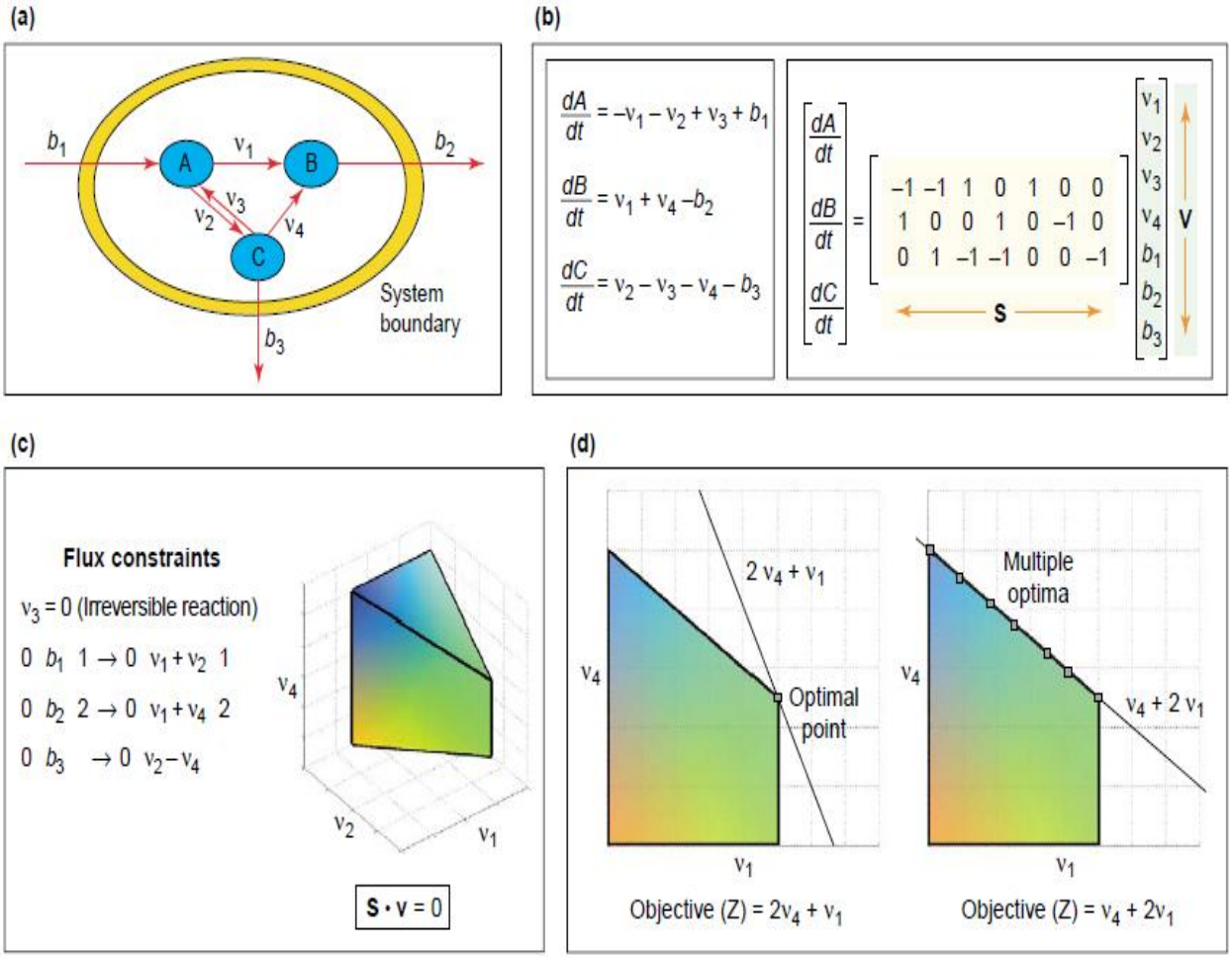


Figure 1 - Overview of FBA. Image adapted from [8].

(a)- Defining the system, (b)- Calculating the reaction stoichiometry,

(c)- Calculating the objective function, (d)- Optimization

(a) **Defining the system:** This includes all the reactions involved in the particular network along with all the co-factors, co-enzymes, reaction reversibility, boundaries of reactions, and compartmentalization.

(b) **Calculating the reaction stoichiometry**: After the definition of all the reactions in the system are set, there is a need to create the stoichiometry matrix that will be used for mass balance after defining the vector of internal fluxes and exchange fluxes.

Under steady state the assumption is

$$dx/dt = S.v = 0$$

Where '***S***' denotes the stoichiometry matrix, '***v***' is the vector of fluxes, and '***dx/dt'*** is the rate of change in concentration.

(c) **Calculating the objective function:** Calculating the objective function and adding more constraints to the system. This is performed after defining the stoichiometry and mass balance is the next step. Here, the system is still incomplete and to complete it we need to add certain constraints to our system, which can be spatial, topological, environmental, and regulatory constraints. The mechanism to add these constraints is to measure certain fluxes and add an upper bound and a lower bound to these fluxes.

(d) **Optimization:** In this step, the steady state mass balance is obtained by using mass balance constraints and other constraints over the objective function defined in the previous step.

1.4 Applications of FBA

FBA has its core applications in (a) identification of possible drug targets, after the metabolic pathway is mapped, its perturbations and simulation is done by FBA and these simulation includes gene knockouts and drug effects [9], also in (b) increasing the biomass of industrially important organisms (such as *E. coli*) after the metabolic network is mapped, certain gene deletions are studied in order to achieve the target [10].

1.5 Short review of existing FBA tools

Many FBA applications, tools, and integrated frameworks in software such as MATLAB are available and here we studied them to find the advantages and disadvantages associated with each of them.

1. COBRA [3] is considered as a good FBA tool. However, as it is a MATLAB [11] toolbox it affects its applicability because not everyone has a commercial license to MATLAB and there is a learning curve in understanding MATLAB.
2. In Systems Biology Research Tool (SBRT) [12], there is a lack of the integration of algorithm processes with a network visualization program.
3. OptFlux [13] lacks flexibility even though it makes use of multiple algorithms.
4. BioSPICE [14] utilizes only two algorithms and flux-balance optimization is not the main focus of this tool.
5. For PathwayAnalyser [15], the installation process is cumbersome with multiple steps and multiple dependencies. It covers only two FBA variants.

Keeping in view of the advantages and disadvantages involved with the available tools and applications, we defined our objective of this study as described further.

1.6 Objective

In this study, we aim to create a web and user friendly tool for carrying out FBA for a given pathway. There are many tools available but they lack in many ways for easy implementation. Thus, we aim to design efficient algorithms to carry out the analysis. Irrespective of the application in mind, we wish to provide a generalized tool for model reconstruction and to carry out FBA.

1.7 Proposed web based tool

The proposed web based tool is called MetFBA (**Met**abolic **F**lux **B**alance **A**nalysis). This web based tool will be freely available to all the users globally. The user will be able to perform FBA on their input metabolic pathway files which will be in standard and most acceptable Systems Biology Markup Language (SBML) or xml formats. There will also be a help page where tutorials about how to perform FBA analysis will be available. The tool wouldn't require any other commercial tool for its functionality. The user will be able to contact us through a form provided in the website, where they will be able to give us feedback and suggestions about the modifications or the updates required for the tool.

The tool is available at: **http://bioinfoindia.org/MetFBA/**

CHAPTER 2 - Materials and Methods

2.1 Description of the Project

Requirement: Requirement of a user friendly web based tool for FBA that provides the user easy functionality to upload and analyze their metabolic network data.

Input: Reactions in a general format like SBML.

Output: Analysis of metabolic pathway according to various constraints that will be set by the user.

2.2 System requirements

Server Machine: Xampp setup or any other option with Apache server.

Client Machine: PC with server connection.

Software: Perl in server machine with CGI compatibility.

2.3 SDLC Model:

A Software Development Life Cycle (SDLC) corresponds to the important phases essential for developers while creating any new software, like planning, analyzing, designing, and implementing. A software development lifecycle covers all the stages of software development from its start with defining the requirement through to maintaining the software.

Multiple SDLC models are available like waterfall, iterative, V shaped, agile, etc. All these models were studied, their properties, advantages, disadvantages were studied and it was concluded that for MetFBA, Waterfall model was the most appropriate one to carry out the project.

Waterfall model (Figure 2) has following characteristics:

- It is a sequential development model.
- In this model, the requirements are required to be clear before going to the next phase.
- Testing is allowed to be carried out only if the entire code is developed.
- The phases of development need to happen one after the next in order and there is no overlap between two phases.
- The work progress should be documented after completion of each phase.
- The testing also happens at the end of each phase. This practice helps with the maintenance of the quality of the project.
- Each step is frozen before the next step, i.e. freeze the requirements beforehand, after that only the coding and other implementation can happen.
- There should be a time limit for the completion of each phase.

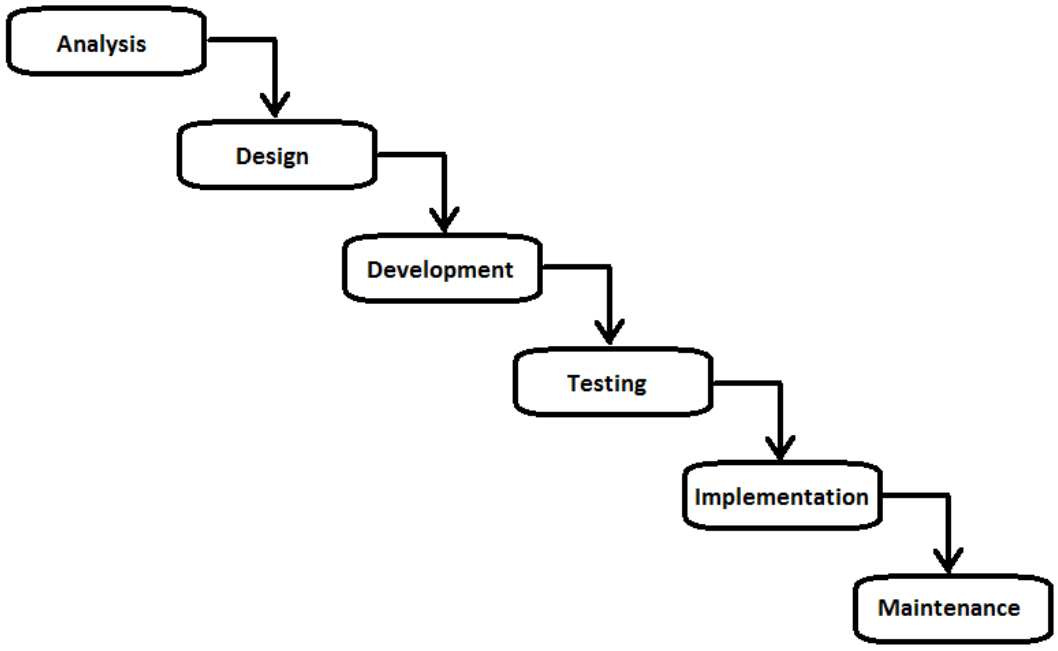


Figure 2 - Waterfall SDLC Model. Image adapted from [16].

In this project, the current steps have been followed:

- First, we analyzed all the information about the FBA, its applications, already existing tools, the implementation of these tools, what gaps need to be filled, how our tool should be different, how it would be able to fill those gaps, and similar information.
- Next, we tried to design a strategy about how the website will be created, what part of the project will be done by which project partner, how the front end will look, how we plan to implement algorithms in the backend, which programming languages and tools would be used, etc.
- Next, in the development step we developed a web page using HTML CGI-Perl, JavaScript and CSS.
- After the development is complete, we tested the tool by comparing the output and time utilized to generate that output in multiple existing tools.
- The next step is implementation or launching our web based tool online.
- Finally we will keep updating our website. For example, if a new algorithm has been reported for FBA, we will try to incorporate that algorithm as well in our website. In simpler words we will try to keep up with the maintenance of the website.

In the next page, the work flowchart is shown that indicates the various steps (Figure 3).

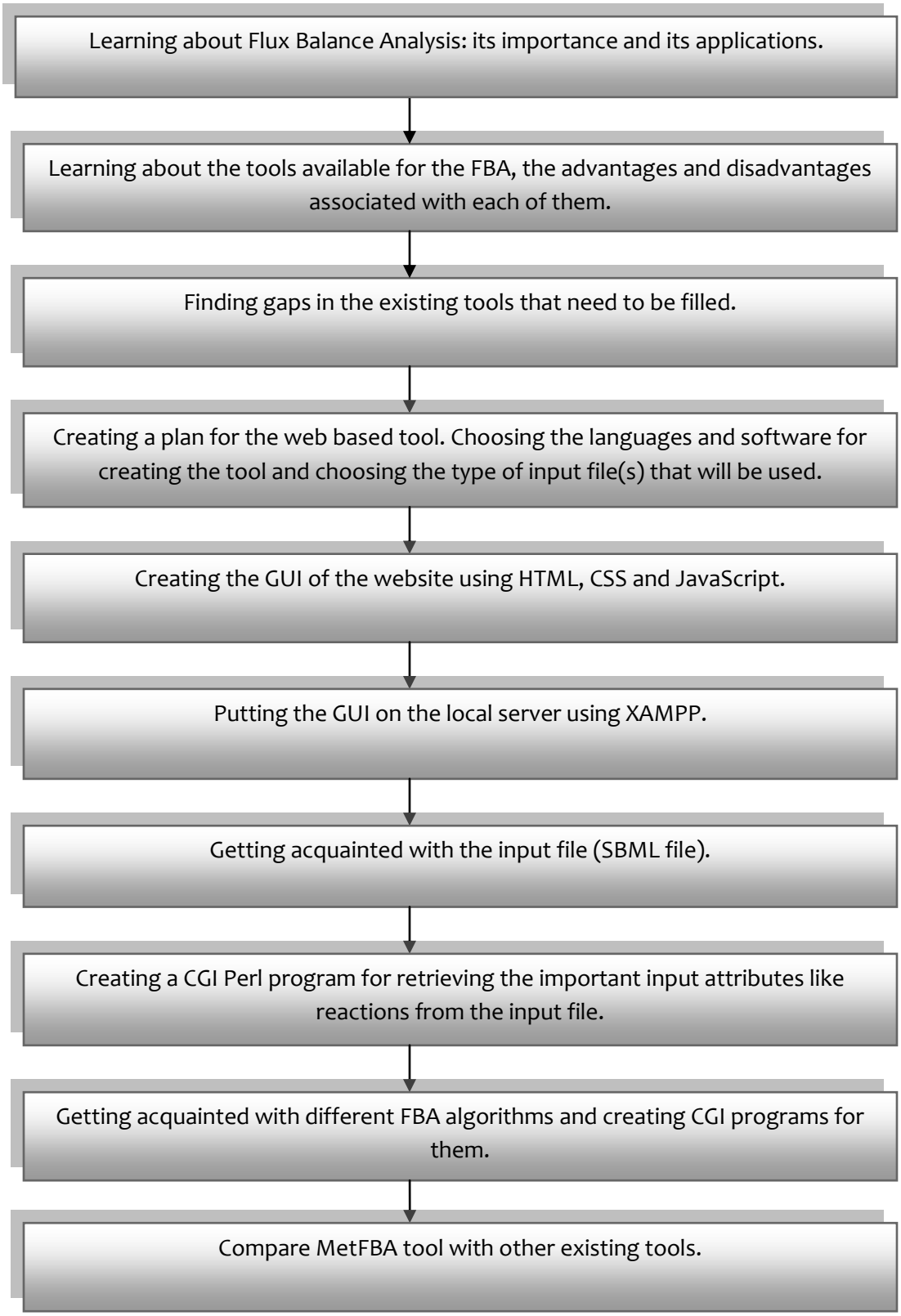


Figure 3 – Workflow

2.4 Programming Languages and tools

To create this tool, multiple tools and languages are going to be required. The choice of these tools and languages was based on their ease of use, how much we were comfortable with them and if they will be able to produce the output that we have in mind.

2.4.1 XAMPP:

XAMPP, developed by Apache Friends, provides a user with all the things required to set up a web server. XAMPP is a cross-platform web server which means it can be used to create a local network on Windows, Mac and Linux equally well. It consists of Apache Server, MariaDB database and interpreters for scripting languages like Perl and PHP (Figure 4). The transition of website from local server to online server is smooth since most web servers make use of same components like XAMPP [17].

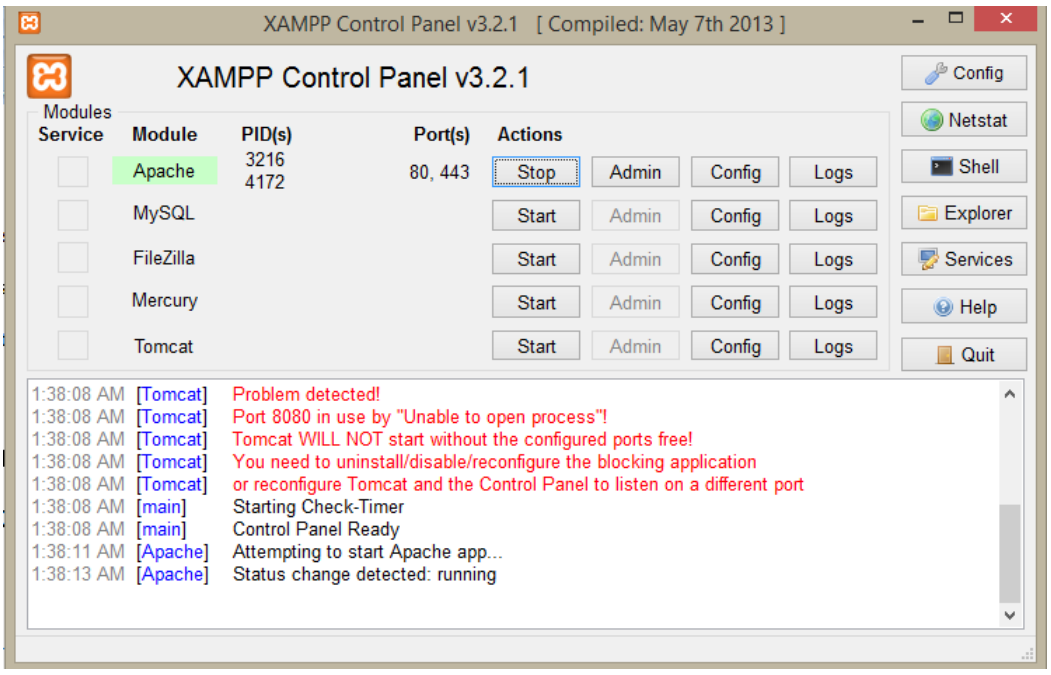


Figure 4 – XAMPP Control Panel

2.4.2 HTML:

HTML is used worldwide as a markup language to create rich webpage and applications. It makes use of tags to create structured webpage, for example, in HTML tables can be created, text can be styled making use of tags like font, lists can be produced, hyperlinks can be provided, images, videos and other objects can be embedded, and much more. HTML can be written in two syntaxes, one of which is HTML and the other one is XML. Even though XML is faster than HTML, but browser support is limited in case of XML, so HTML will be preferred over XML for this project [18].

2.4.3 JavaScript

JavaScript is a scripting language that is used to make dynamic web pages. It works in collaboration with HTML and is used majorly to create responsive web pages for example moving images, responsive buttons, slideshows, etc.

2.4.4 Cascading Style Sheets (CSS)

CSS is a style sheet language that is used to set the appearance of a markup language in our case HTML. Along with HTML and JavaScript, CSS is the third language used for creating web pages. CSS is majorly used to separate the web page content from its presentation so that one can generally set the presentation of documents instead of configuring it every time when adding or changing some content.

2.5 Optimization techniques used by different tools

Other tools or software that works on metabolic networks modeling of FBA uses different packages for optimization some of these are given in Table 2.

Table 2 – Optimization techniques used by existing FBA tools

Software/Tool	Reference	Optimization Solver	Reference
OptFlux	[13]	GLPK	[19]
SBRT	[12]	GLPK	[19]
MetaFluxNet	[20]	LP_SOLVE	[25]
BioMet	[21]	GLPK	[19]
SurreyFBA	[22]	GLPK	[19]
FASIMU	[23]	GLPK, CPLEX, LP_SOLVE	[19, 24, 25]
COBRA Toolbox	[3]	GLPK, TOMLAB_CPLEX	[19, 24]
CellNetAnalyzer	[26]	GLPK, Optimization Toolbox (MATLAB)	[19, 11]

The solver used in our tool for optimization is GNU Linear Programming Kit (GLPK) implemented in JavaScript.

2.5.1 GNU Linear Programming Kit (GLPK) for JavaScript

GLPK is a open source optimization problem solver that is provided by GNU and can handle very large problems. We have used its extension for JavaScript which was available on GitHub (https://github.com/hgourvest/glpk.js). Compared to the other linear programming options which are shown in Table 2, we used GLPK as its integration to a web based tool is easier. .

CHAPTER 3 – Results

3.1 GUI of MetFBA:

For the creation of Graphical User Interface (GUI) the code was written in the HTML and the text was styled using CSS. The pages were made dynamic using JavaScript. In the home page, the user is asked to input a SBML (.sbml file extension) file. The user can then browse through his computer and upload the SBML file. The default upper and lower bound values can also be provided in the form. The user can then submit the form after entering all the required values.

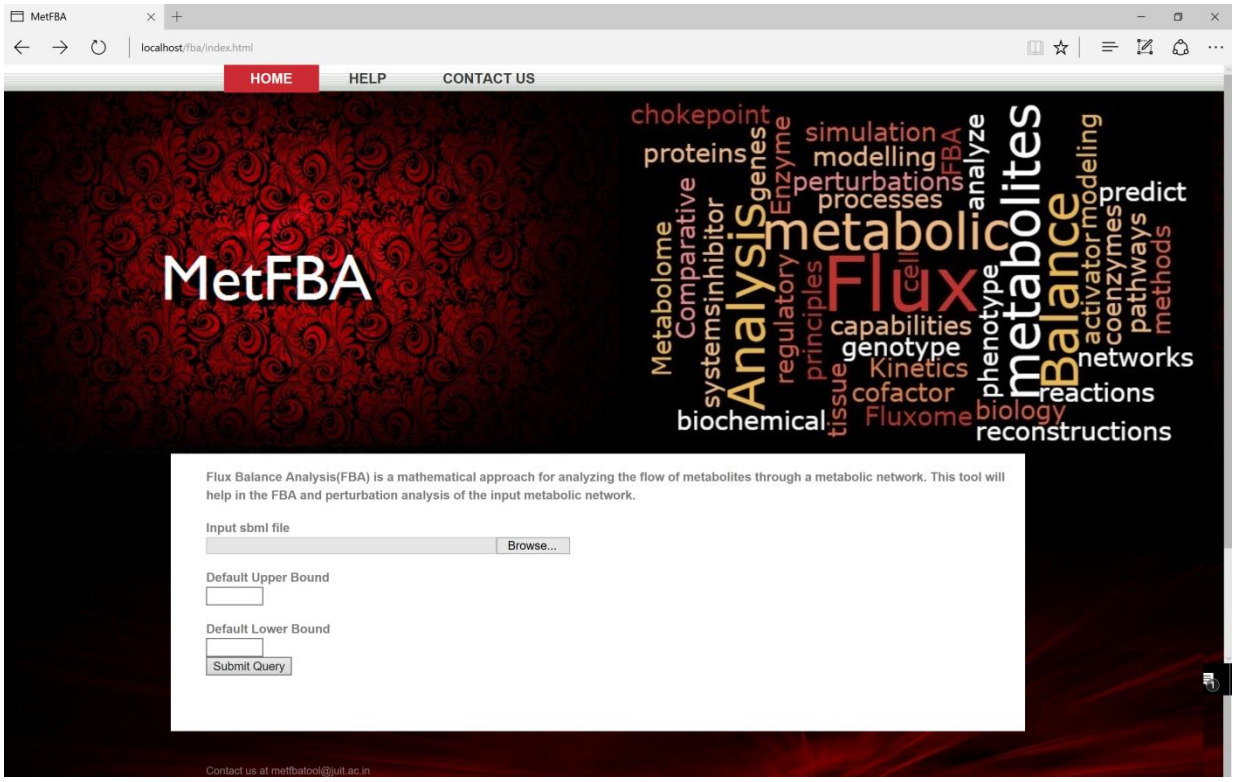


Figure 5 – Screenshot of the home page of MetFBA

In the Help tab, the users will be provided with a tutorial for the analysis to make their experience easier with the tool.

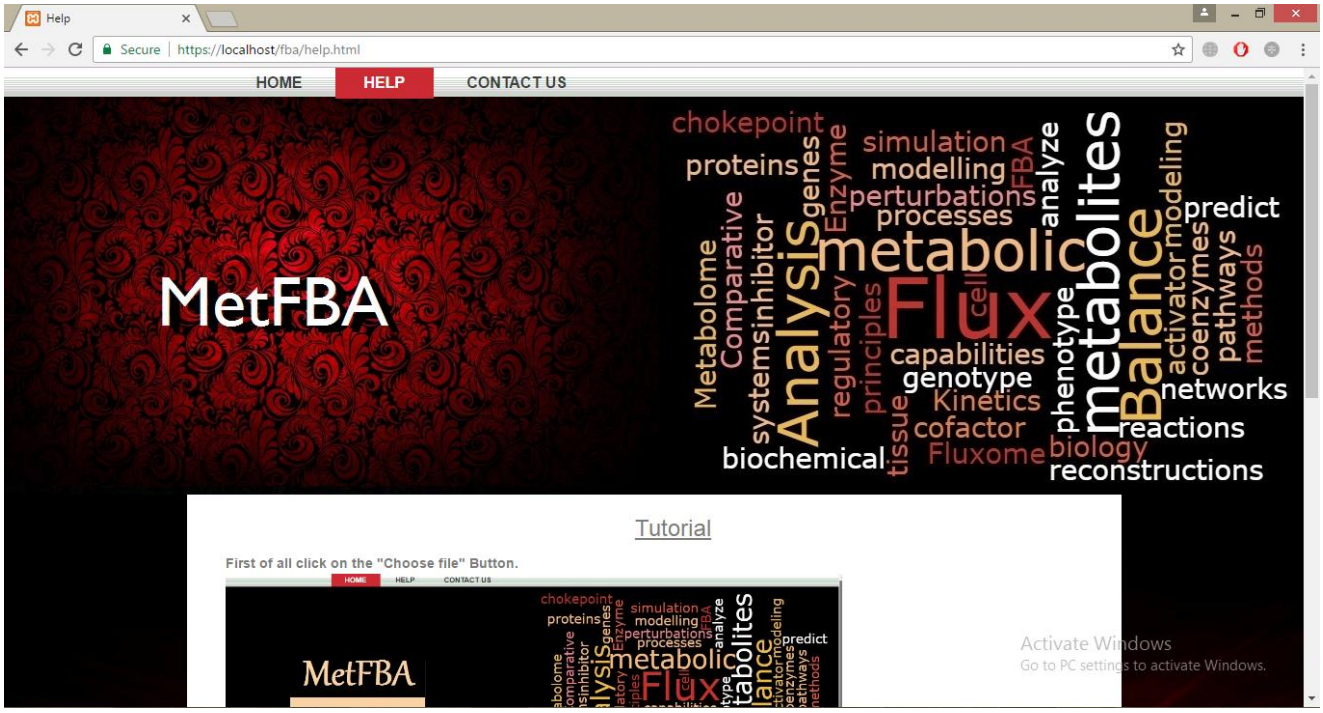


Figure 6 - Help Tab

In the Contact Us page, one can reach out to us and give us feedback and suggestions by filling up the form there.

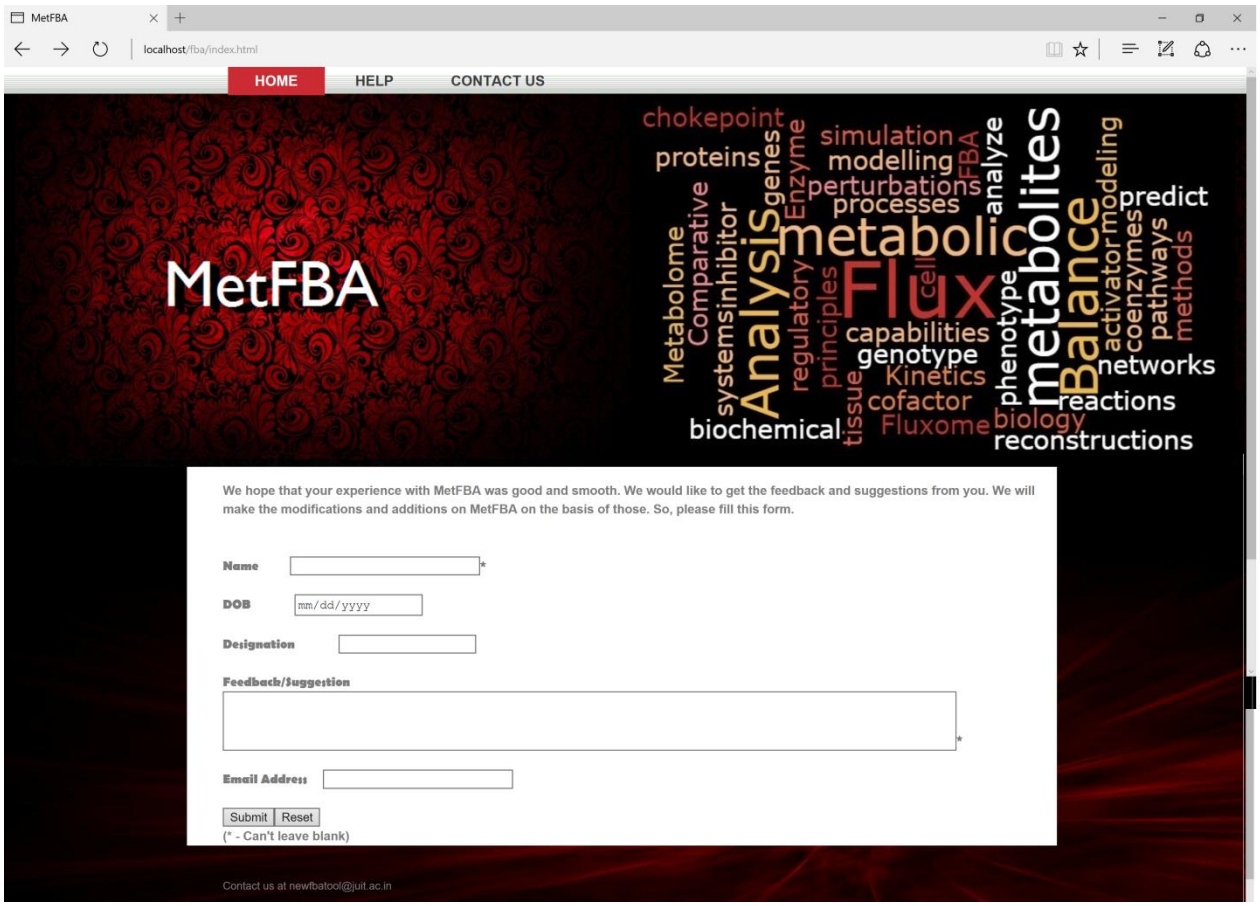


Figure 7 - Contact us Tab

3.2 Perl - CGI:

The code for reading the SBML file and retrieving the important input parameters for FBA, like reactions' name, reactions' id, reactions with the metabolite name and metabolite ids, was created simply by using perl programming language (code in Appendix-I). The values were retrieved into different variables and were printed into different files so that they can be used in the later analyses. When a sample SBML file (example below) was provided as input, the given output (Figure 8) was retrieved.

Sample SBML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2"
version="1">
<model name="Test">
<listOfCompartments>
<compartment id="cell" size="1" name="cell interior" />
<compartment id="ext" size="1" name="blood compartment" />
</listOfCompartments>
<listOfSpecies>
<species id="Glc_ext"  name="Glucose_ext"
boundaryCondition="true"  compartment="ext" />
<species id="Pi_ext"  name="Phosphate_ext"
boundaryCondition="true"  compartment="ext" />
.
.
</listOfSpecies>
<listOfReactions>
<reaction id="GlcT" name="GlcT" reversible="false">
<listOfReactants>
<speciesReference species="Glc_ext" stoichiometry="1"/>
</listOfReactants>
<listOfProducts>
<speciesReference species="Glc_cell" stoichiometry="1"/>
</listOfProducts>
<kineticLaw>
<listOfParameters>
<parameter id="equilibrium_constant" value="1"/>
</listOfParameters>
</kineticLaw>
</reaction>
<reaction id="HK" name="HK" reversible="true">..</reaction>
.
.
</listOfReactions>
</model>
</sbml>
```

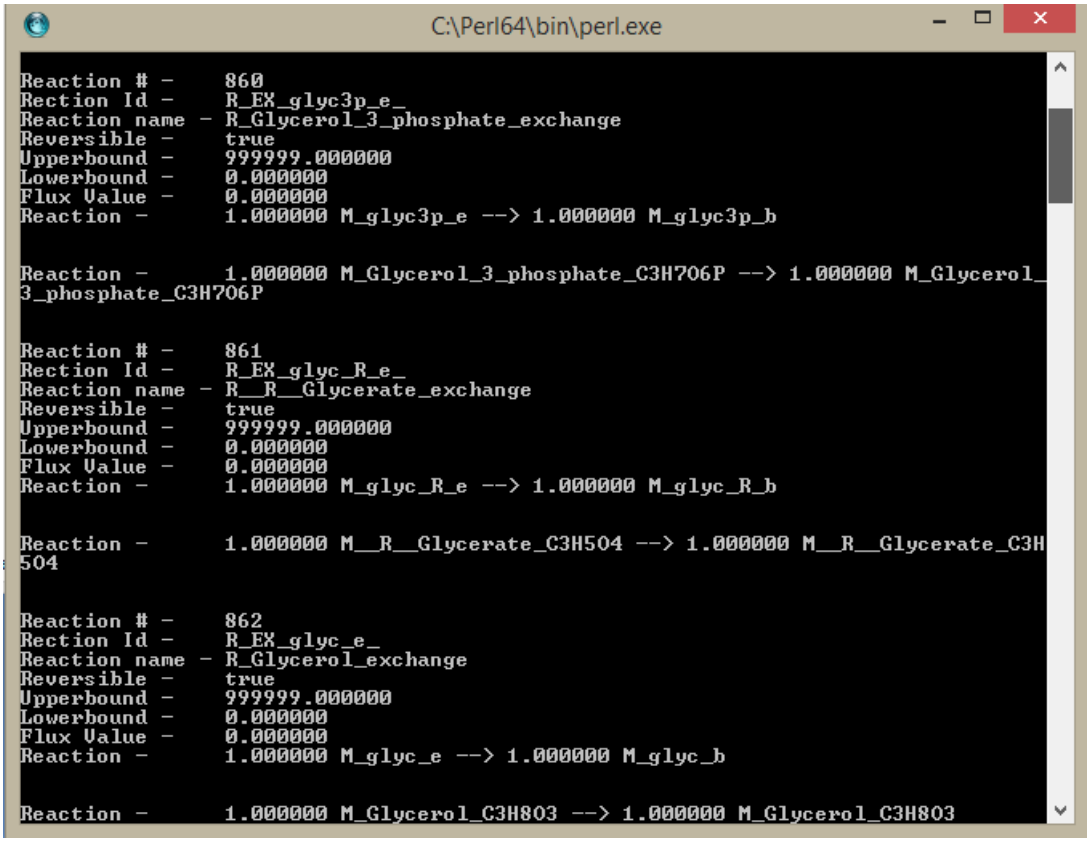


Figure 8 – Console output

The perl code for parsing input file was used as CGI Perl program to utilize the code in the server. When the user inputs SBML file in the form present in the home page of the website (Figure 5), the website redirects to the CGI script link and CGI Perl code uses that file as input and produces output on the webpage (Figure 9). Multiple text files are also created for use in the subsequent steps. They are: “reactions.txt” in which all the reactions are stored, “speciesid.txt” for the species ids, “reactionid.txt” for the reaction ids, “objective_function.txt” for the objective function if the values of fluxes are available in the input file and “bounds.txt” for the upper and lower bounds associate with each reaction.

“reactions.txt” file generated for the sample SBML file:

```
1.000000 M_12dgr120_p -->  1.000000 M_12dgr120_c
1.000000 M_12dgr140_p -->  1.000000 M_12dgr140_c
1.000000 M_12dgr141_p -->  1.000000 M_12dgr141_c
1.000000 M_12dgr160_p -->  1.000000 M_12dgr160_c
1.000000 M_12dgr161_p -->  1.000000 M_12dgr161_c
1.000000 M_12dgr180_p -->  1.000000 M_12dgr180_c
1.000000 M_12dgr181_p -->  1.000000 M_12dgr181_c
1.000000 M_12ppd_R_e -->  1.000000 M_12ppd_R_p
1.000000 M_12ppd_R_p -->  1.000000 M_12ppd_R_c
1.000000 M_12ppd_S_e -->  1.000000 M_12ppd_S_p
1.000000 M_12ppd_S_p -->  1.000000 M_12ppd_S_c
1.000000  M_atp_c  +  1.000000  M_h2o_c  +  1.000000  M_14glucan_p  -->   1.000000
M_14glucan_c +  1.000000 M_adp_c +  1.000000 M_h_c +  1.000000 M_pi_c
1.000000 M_14glucan_e -->  1.000000 M_14glucan_p.
.
.
.
1.000000 M_zn2_e -->  1.000000 M_zn2_p
```

“speciesid.txt” file generated for the sample SBML file:

```
M_10fthf_c
M_12dgr120_c
M_12dgr120_p
M_12dgr140_c
M_12dgr140_p
M_12dgr141_c
M_12dgr141_p
M_12dgr160_c
M_12dgr160_p
M_12dgr161_c
M_12dgr161_p
M_12dgr180_c
M_12dgr180_p
.
.
.
M_zn2_b
```

“reactionid.txt” generated for the sample SBML file:

```
R_12DGR120tipp
R_12DGR140tipp
R_12DGR141tipp
R_12DGR160tipp
```

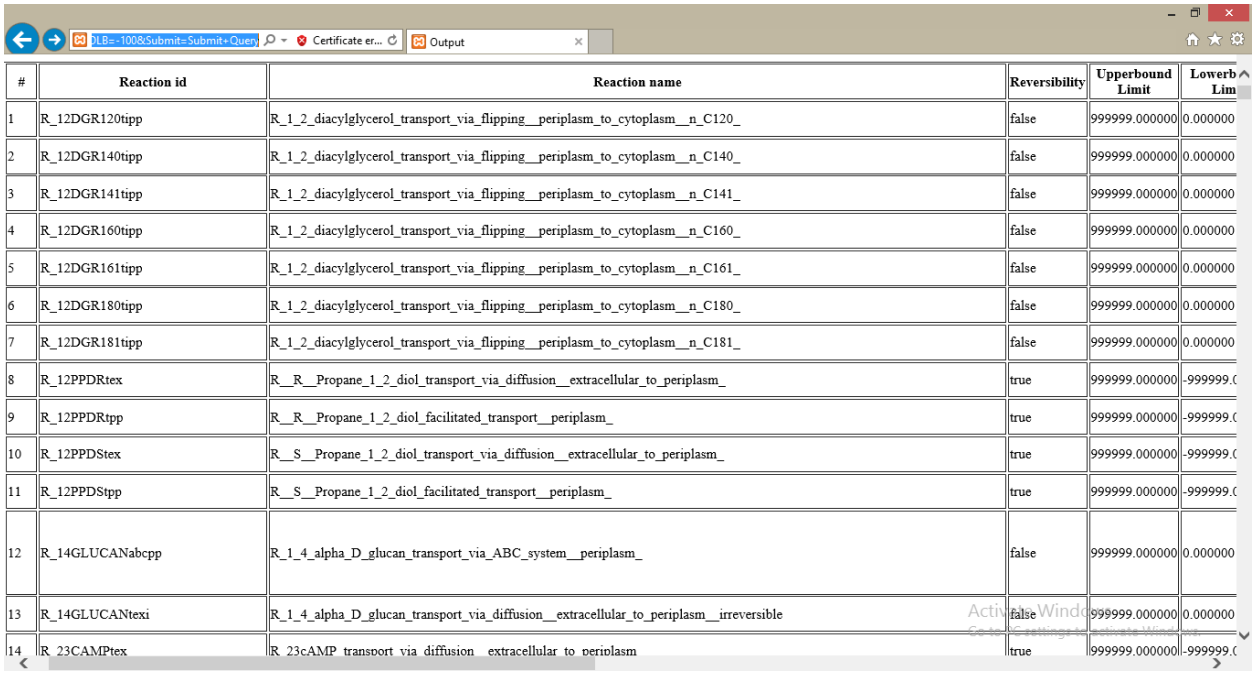
```
R_12DGR161tipp
R_12DGR180tipp
R_12DGR181tipp
R_12PPDRtex
R_12PPDRtpp
R_12PPDStexR_12DGR120tipp
R_12DGR140tipp
.
.
.
R_Zn2tex
```

"objective_function.txt" generated for the sample SBML file:

```
+0.284212   v(R_3HAD100)   +0.164070   v(R_3HAD120)   +0.120141   v(R_3HAD121)
+0.101931   v(R_3HAD140)   +0.120141   v(R_3HAD141)   +0.101931   v(R_3HAD160)
+0.120141   v(R_3HAD161)    +0.284212    v(R_3HAD40)    +0.284212    v(R_3HAD60)
+0.284212   v(R_3HAD80)    +0.284212   v(R_3OAR100)    +0.164070   v(R_3OAR120)
+0.120141   v(R_3OAR121)   +0.164070   v(R_3OAR140)   +0.120141   v(R_3OAR141)
+0.101931   v(R_3OAR160)   +0.120141   v(R_3OAR161)   +0.284212   v(R_3OAR40)
+0.284212   v(R_3OAR60)    +0.284212   v(R_3OAR80)    +0.284212   v(R_3OAS100)
+0.164070   v(R_3OAS120)   +0.120141   v(R_3OAS121)   +0.164070   v(R_3OAS140)
+0.120141   v(R_3OAS141)   +0.101931   v(R_3OAS160)   +0.120141   v(R_3OAS161)
+0.284212   v(R_3OAS60)
+0.284212   v(R_3OAS80) +0.031070 v(R_A5PISO)
+......  +0.002522 v(R_Zn2tex)
```

"bounds.txt" generated for the sample SBML file:

```
v(R_12DGR120tipp) < 999999.000000
v(R_12DGR120tipp) > 0.000000
v(R_12DGR140tipp) < 999999.000000
v(R_12DGR140tipp) > 0.000000
v(R_12DGR141tipp) < 999999.000000
v(R_12DGR141tipp) > 0.000000
v(R_12DGR160tipp) < 999999.000000
v(R_12DGR160tipp) > 0.000000
v(R_12DGR161tipp) < 999999.000000
v(R_12DGR161tipp) > 0.000000
v(R_12DGR180tipp) < 999999.000000
v(R_12DGR180tipp) > 0.000000
.
.
.
v(R_Zn2tex) > -999999.000000
```

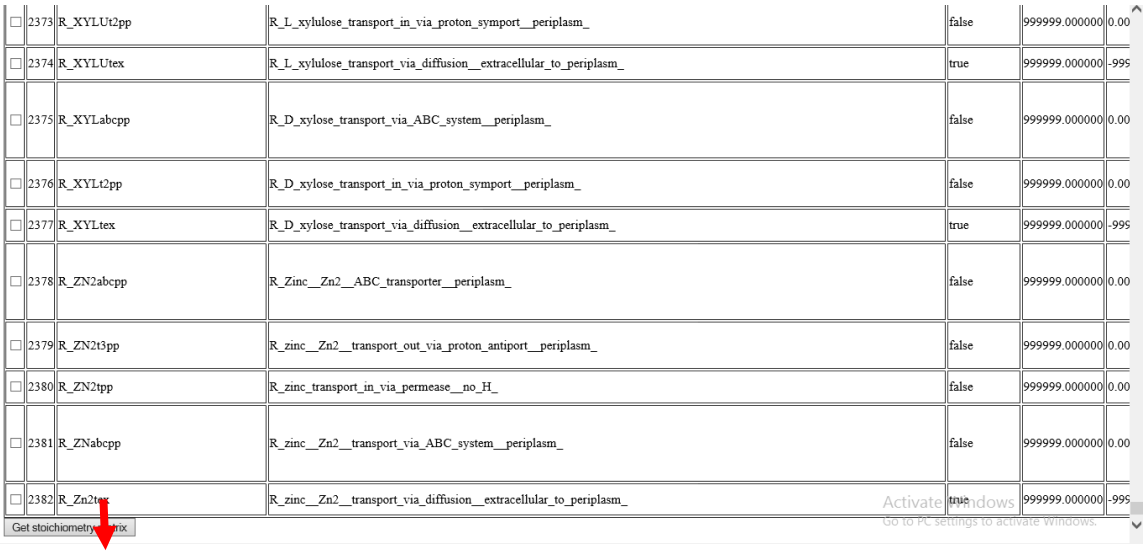


#	Reaction id	Reaction name	Reversibility	Upperbound Limit	Lowerb Lim
1	R_12DGR120tipp	R_1_2_diacylglycerol_transport_via_flipping__periplasm_to_cytoplasm__n_C120_	false	999999.000000	0.000000
2	R_12DGR140tipp	R_1_2_diacylglycerol_transport_via_flipping__periplasm_to_cytoplasm__n_C140_	false	999999.000000	0.000000
3	R_12DGR141tipp	R_1_2_diacylglycerol_transport_via_flipping__periplasm_to_cytoplasm__n_C141_	false	999999.000000	0.000000
4	R_12DGR160tipp	R_1_2_diacylglycerol_transport_via_flipping__periplasm_to_cytoplasm__n_C160_	false	999999.000000	0.000000
5	R_12DGR161tipp	R_1_2_diacylglycerol_transport_via_flipping__periplasm_to_cytoplasm__n_C161_	false	999999.000000	0.000000
6	R_12DGR180tipp	R_1_2_diacylglycerol_transport_via_flipping__periplasm_to_cytoplasm__n_C180_	false	999999.000000	0.000000
7	R_12DGR181tipp	R_1_2_diacylglycerol_transport_via_flipping__periplasm_to_cytoplasm__n_C181_	false	999999.000000	0.000000
8	R_12PPDRtex	R__R__Propane_1_2_diol_transport_via_diffusion__extracellular_to_periplasm_	true	999999.000000	-999999.(
9	R_12PPDRtpp	R__R__Propane_1_2_diol_facilitated_transport__periplasm_	true	999999.000000	-999999.(
10	R_12PPDStex	R__S__Propane_1_2_diol_transport_via_diffusion__extracellular_to_periplasm_	true	999999.000000	-999999.(
11	R_12PPDStpp	R__S__Propane_1_2_diol_facilitated_transport__periplasm_	true	999999.000000	-999999.(
12	R_14GLUCANabcpp	R_1_4_alpha_D_glucan_transport_via_ABC_system__periplasm_	false	999999.000000	0.000000
13	R_14GLUCANtexi	R_1_4_alpha_D_glucan_transport_via_diffusion__extracellular_to_periplasm__irreversible	false	999999.000000	0.000000
14	R_23CAMPtex	R_23cAMP_transport_via_diffusion__extracellular_to_periplasm	true	999999.000000	-999999.(

Figure 9 – Output on the webpage

In the example, there are 2382 reactions.

At the bottom of this page, there is a button called “Get stoichiometric matrix” (Figure 10). On clicking that button, we will get the stoichiometric matrix created on the basis of reactions and metabolites present in this model.



☐	2373	R_XYLUt2pp	R_L_xylulose_transport_in_via_proton_symport__periplasm_	false	999999.000000	0.00
☐	2374	R_XYLUtex	R_L_xylulose_transport_via_diffusion__extracellular_to_periplasm_	true	999999.000000	-999
☐	2375	R_XYLabcpp	R_D_xylose_transport_via_ABC_system__periplasm_	false	999999.000000	0.00
☐	2376	R_XYLt2pp	R_D_xylose_transport_in_via_proton_symport__periplasm_	false	999999.000000	0.00
☐	2377	R_XYLtex	R_D_xylose_transport_via_diffusion__extracellular_to_periplasm_	true	999999.000000	-999
☐	2378	R_ZN2abcpp	R_Zinc__Zn2__ABC_transporter__periplasm_	false	999999.000000	0.00
☐	2379	R_ZN2t3pp	R_zinc__Zn2__transport_out_via_proton_antiport__periplasm_	false	999999.000000	0.00
☐	2380	R_ZN2tpp	R_zinc_transport_in_via_permease__no_H_	false	999999.000000	0.00
☐	2381	R_ZNabcpp	R_zinc__Zn2__transport_via_ABC_system__periplasm_	false	999999.000000	0.00
☐	2382	R_Zn2tex	R_zinc__Zn2__transport_via_diffusion__extracellular_to_periplasm_	true	999999.000000	-999

Get stoichiometry matrix

Figure 10 - Button to get stoichiometric matrix

Using *reactions.txt* and *speciesid.txt*, stoichiometric matrix is created. Another perl code (Appendix-II) was created to obtain the stoichiometric matrix and was then used as CGI Perl code. When the user clicks the button "Get stoichiometric matrix" (Figure 10), the webpage redirects to that CGI Perl link and the script uses the two files as the input stores the matrix as *stoichiometric.txt*. The user will be redirected to the given page.(Figure 11).

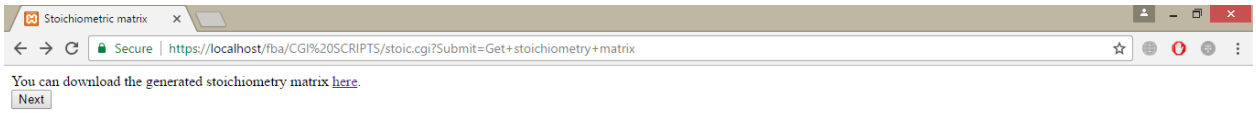


Figure 11 – Stoichiometric Matrix can be downloaded on clicking the hyperlink "here"

"stoichiometric.txt":

```
0       0       0       0       0       0       .       .       .       .       .       0
0       0       0       0       0       0       .       .       .       .       .       0
0       0       0       0       0       0       .       .       .       .       .       0
0       0       0       0       0       0       .       .       .       .       .       0
0       0       0       0       0       0       .       .       .       .       .       0
1.000000        0       0       0       0       0       .       .       .       .       .       0
.       .       .       .       .       .       .       .       .       .       .       0
.       .       .       .       .       .       .       .       .       .       .       0
0       0       0       0       0       0       0       .       .       .       .       0
```

The sample input file had 2382 reactions and 1972 species ids, therefore this matrix is 1972 x 2382 dimensional.

On clicking, "Next" in Figure 11, the user will be redirected to a page where using perl script (Appendix-III) the linear programming problem(LPP), containing the objective function which is either to be maximized or minimized, variable constraints and boundaries, is generated from

the given input. The script uses "reactionid.txt", “stoichiometric.txt", "bounds.txt", objective_function.txt” created earlier to generate the LPP (Figure 12, Figure 13, and Figure 14). The flux values for multiple reactions were available for the sample input file and thus objective function was retrieved using those values.

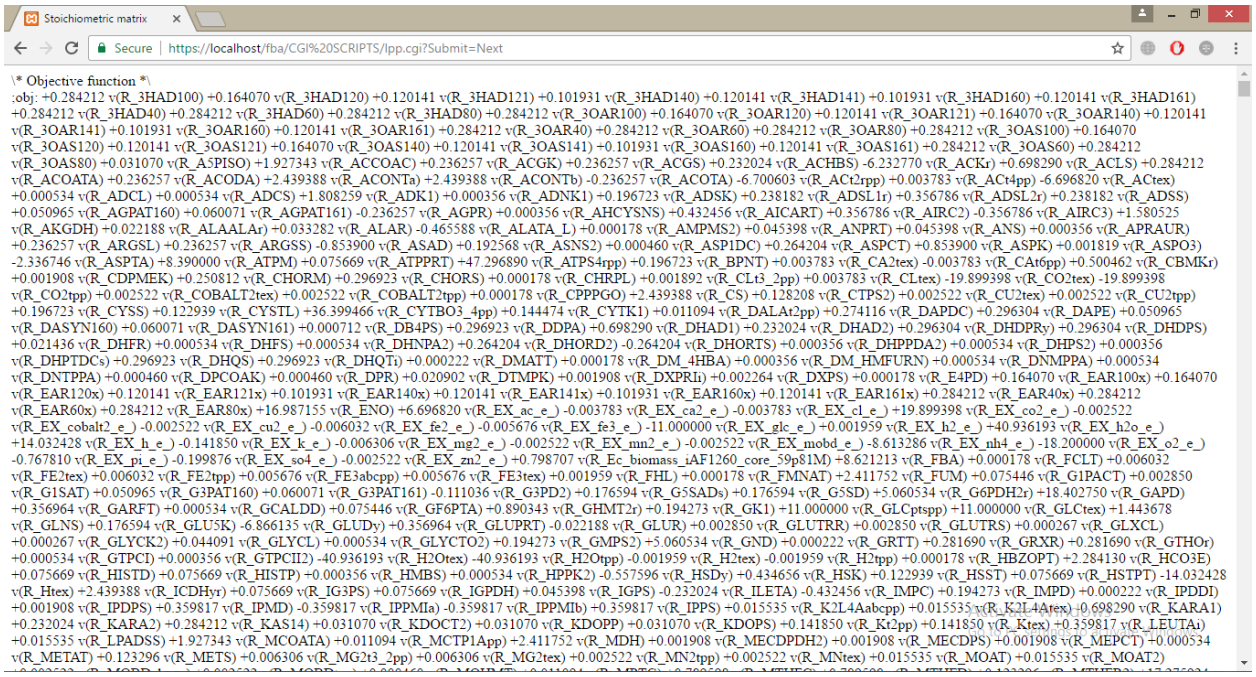


Figure 12 – Objective Function

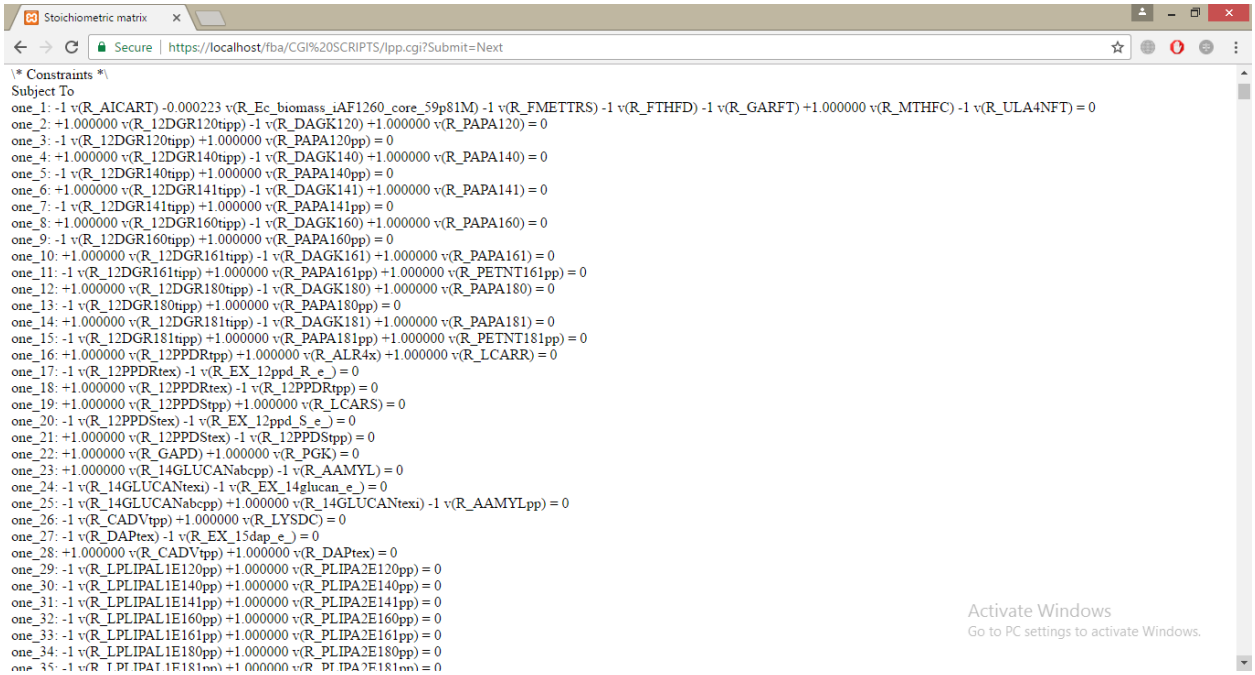


Figure 13 – Constraints

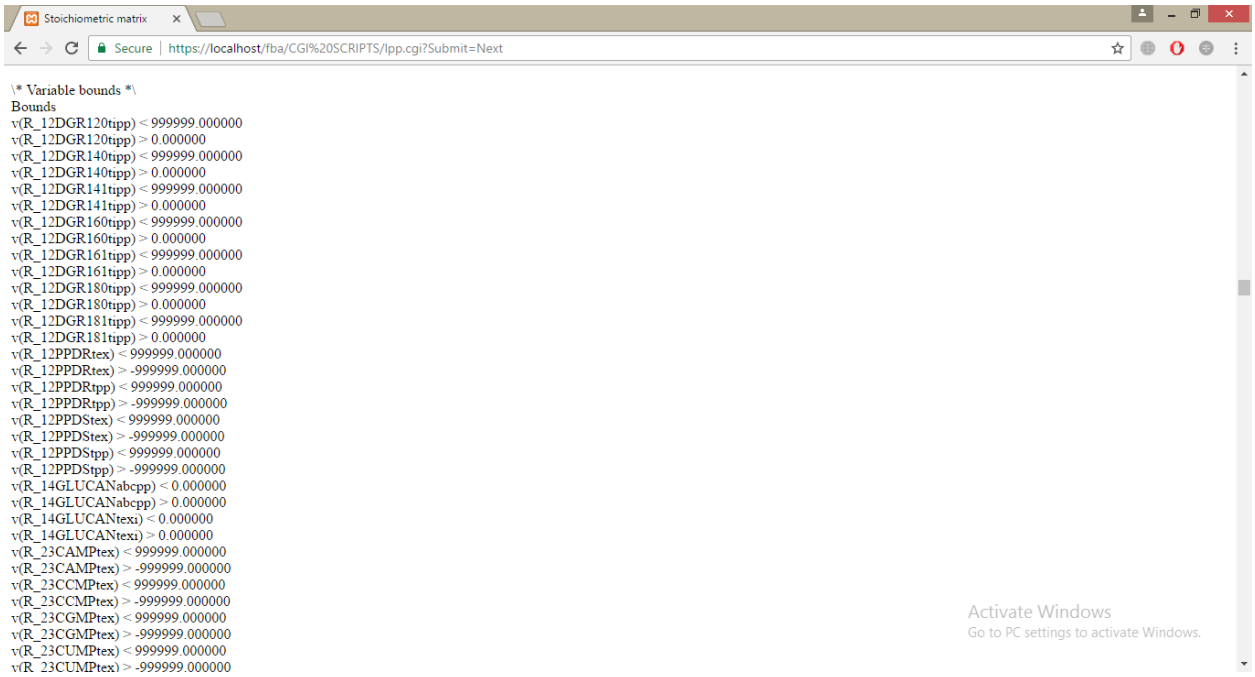


Figure 14 - Bounds

At the bottom of the page, “Next” button is present. On clicking that, user will be redirected to a page (Figure 15) where optimization can be performed using GLPK. In the case of flux values are not available in the input file; users can enter their own objective functions in the given textbox or can edit the objective function created by the MetFBA using the input file. They can also make changes to constraints and bounds. On clicking “Submit”, the optimized output will be generated (Figure 16).

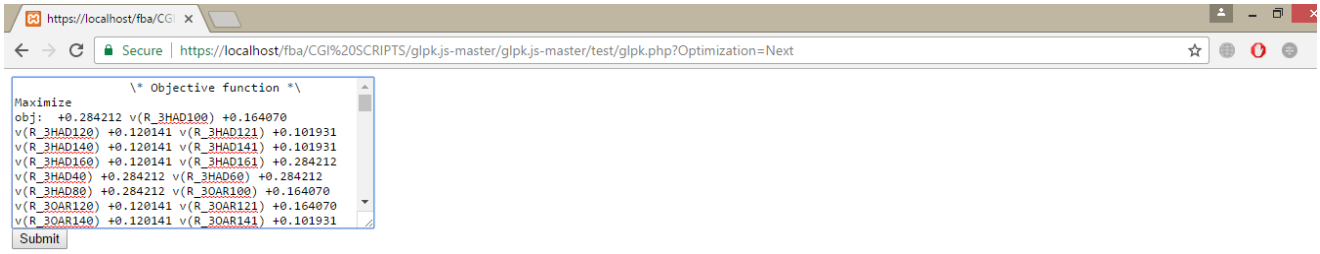


Figure 15 – GLPK interface

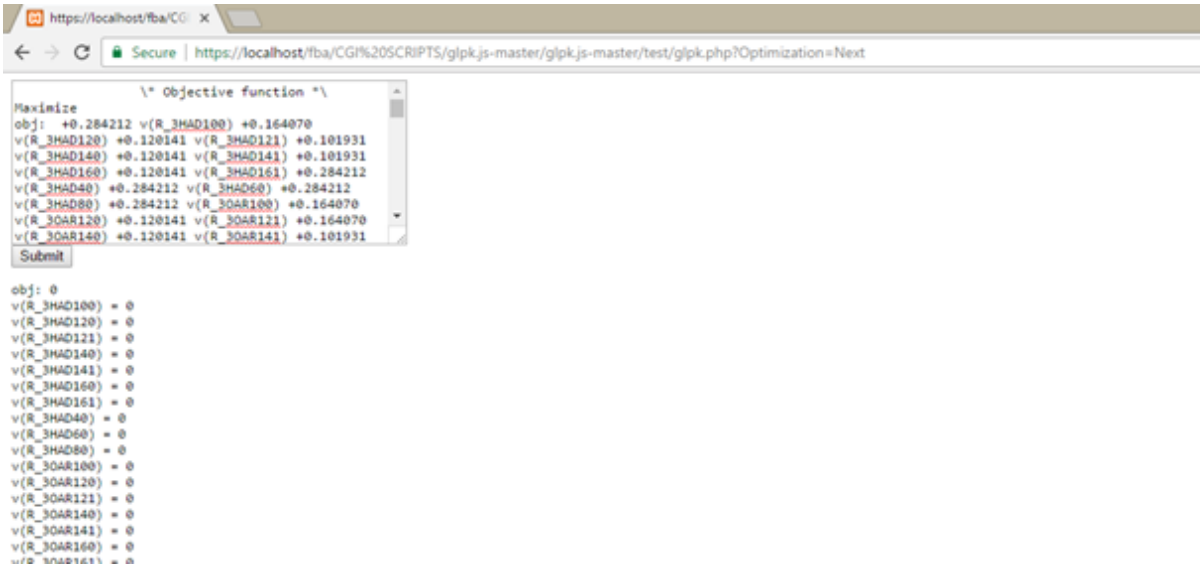


Figure 16 – Optimized Values for objective function and the variables

3.4 Comparing MetFBA with existing tools/software

The output generated by MetFBA was compared with already existing tools in terms of output for the same objective function and also in terms of time utilized in the execution. Here, we used three different objective functions using three different input .sbml files. For conveience sake, we have listed them as Input I, Input II, and Input III and the outputs (Output I, Output II, and Output III) and the time utilized (Time Utilized I, Time Utilized II, and Time Utilized III), repectively. The comparison is tabulated in Table 3.

Table 3 – Comparison by different tools

Tool	Output I	Time Utilized I	Output II	Time Utilized II	Output III	Time Utilized III
MetFBA	0	20 minutes	0	20 minutes	9.999	2 minutes
Optflux	0.00006	41 minutes	0.00003	38 minutes	8.765	4 minutes
MetExplore(Uses SurreyFBA)	0.917246	2 minutes	0.736701	4 minutes	9.999	1 minute
BioMet FBA	Output generated in terms of fluxes.	6 minutes	Output generated in terms of fluxes.	6 minutes	Output generated in terms of fluxes.	1minute

The output of MetFBA consisted of the optimized value of the objective function and the value associated with each of the reaction fluxes. In comparison, Optflux gave the outputs similar to the outputs generated by MetFBA, but took more time compared to MetFBA. Metexplore, in contrast, gave the value for the optimized objective function but not of the individual reaction fluxes. In case of BioMet, it does not let the user make changes or create an objective function, thus it does not provide the optimized value of the objective function. BioMet gave the final output in terms of whether a reaction can carry a flux based on the positive and negative value of fluxes. We encountered either installation or compilation issues with SBRT, COBRA, and Pathway Analyze, thus we were unable to compare MetFBA with these tools. The web based tool, MetaFluxNet, is no longer available online, thus unable to compare its results.

Chapter 4 – CONCLUSIONS

We have successfully implemented a web based tool called MetFBA, where a user can submit a .sbml file to carry out Flux Balance Analysis (FBA). The current tool is user-friendly and results are generated in a reasonable amount of time. Comparing MetFBA with other existing tools that perform FBA, we conclude that MetFBA gives comparable results for a given objective function. Compared to other tools, we have implemented GNU Linear Programming Kit to either maximize of minimize the objective function.

Currently, the objective function is provided by the user in the .sbml file. Thus, limiting it to perform only on the given objective function, which we hope can be removed by implementing a *de novo* approach of designing an objective function for a given network or reactions in a .sbml file.

Thus, the future work may involve:

- *De novo* generation of objective function.
- Analysis of important genes that are involved in natural product biosynthesis, where optimizing the cellular network for overproduction of natural product will have industrial and commercial application.
- Integrating the network visualization for the given reaction, uploaded by the user.

Appendix - I

Perl Script for input retrieval:

```
#Input Retreival from sbml and XML files, Authors- Shubhangi Kaushik, Devesh Thapar
Version- 1.5 Date of Creation- 14-9-2016
#!

use strict;
use warnings;
use Class::Struct;
use Data::Dumper;

open f1,"13059_2009_2219_MOESM4_ESM.xml"; #Input File
open(f2,">","reactions.txt"); #Reactions will be written in this file
open(f3,">","speciesid.txt"); #Species ids will be written in this file
open(f5,">","reactionid.txt"); #Reaction ids will be written in this file
open(f6,">","objective_function.txt"); #Objective function will be written in this
file
open(f7,">","bounds.txt"); #Reaction boundaries will be written in this file
my (@rxn,$y,@temp,@s,@r,@p,@sid,@ab,@ab2,@ab3,@ab4,$a8,$b8,$c8,$x,%s_id,$fv,$u,$l);
my ($flag,$c,$cr,$cp,$s_id,$f2,$f3,$count)=(0,0,0,0,0,0,0,0);
struct
reaction=>[id=>'$',name=>'$',rev=>'$',r=>'@',s_r=>'@',p=>'@',s_p=>'@',par=>'%',flux_va
l=>'$',ub=>'$',lb=>'$',];

foreach $x(<f1>)
{
$x =~ s/[\"]//g;
       if($x =~ m/^\s*<species id/) #Retreiving species ids and adding them to
speciesid.txt
       {
              $s_id++;
              my@temp= split(" ",$x);
              my@temp2=split("=",$temp[1]);
              print  f3 $temp2[1]."\n";
       }
       #Creating hash for species ids with s_id as keys and name as values
       if($x =~ m/^\s*<species /)
       {
              @temp= split(" ",$x);
              my (@temp2,@temp3);
              for(my $i=0; $i<scalar(@temp); $i++)
              {
                     my @t=split("=",$temp[$i]);
                     if($t[0] eq "id")
                     {
                            @temp2=@t;
                     }
                     elsif($t[0] eq "name")
                     {
                            @temp3=@t;
                     }
```

```
	}
	$s_id{$temp2[1]}=$temp3[1];
}
#reading reactions
if($x =~ m/^\s*<reaction /)
{
	$flag=1;
	$count++;
}
if($flag==1)
{
	if($x =~ m/^\s*<reaction /)
	{
	$x =~ s/[\/\>]//g;

		@temp= split(" ",$x);
		my (@temp2,@temp3,@temp4);
		for(my $i=0; $i<scalar(@temp); $i++)
		{
			my @t=split("=",$temp[$i]);
			if($t[0] eq "id")
			{
				@temp2=@t;
			}
			elsif($t[0] eq "name")
			{
				@temp3=@t;
			}
			elsif($t[0] eq "reversible")
			{
				@temp4=@t;
			}
		}
		$a8=$temp2[1];#reaction id is stored in variable $a8
		$b8=$temp3[1];#reaction name is stored in variable $b8
		$c8=$temp4[1];#reaction reversibility is stored in variable $c8
	}
	if($x =~ m/^\s*<listOfReactants>/)
	{
		$f2=1;
		$cr=0;#count of reactants
		@ab=();
		@ab2=();
		next;
	}
	if($x =~ m/^\s*<\/listOfReactants>/)
	{
		$f2=0;
	}
	if($f2==1)
	{
		$x =~ s/[\/\>]//g;
		@temp= split(" ",$x);
		my (@temp2,@temp3,@temp4);
		for(my $i=0; $i<scalar(@temp); $i++)
```

```
	{
		my @t=split("=",$temp[$i]);
		if($t[0] eq "species")
		{
			@temp2=@t;
		}
		elsif($t[0] eq "stoichiometry")
		{
			@temp3=@t;
		}
	}
	$ab[$cr]=$temp2[1];#reactant(species id) stored in array ab
	$ab2[$cr]=$temp3[1];#stoiciometry of reactant stored in array ab2
	$cr++;
}
if($x =~ m/^\s*<listOfProducts>/)
{
	$f3=1;
	$cp=0;#count of products
	@ab3=();
	@ab4=();
	next;
}
if($x =~ m/^\s*<\/listOfProducts>/)
{
	$f3=0;
}
if($f3==1)
{
	$x =~ s/[\/\>]//g;
	@temp= split(" ",$x);
	my (@temp2,@temp3,@temp4);
	for(my $i=0; $i<scalar(@temp); $i++)
	{
		my @t=split("=",$temp[$i]);
		if($t[0] eq "species")
		{
			@temp2=@t;
		}
		elsif($t[0] eq "stoichiometry")
		{
			@temp3=@t;
		}
	}
	$ab3[$cp]=$temp2[1];#product(species id) stored in array ab3
	$ab4[$cp]=$temp3[1];#stoiciometry of products stored in array ab4
	$cp++;
}
if($x =~ m/^\s*<parameter id=LOWER_BOUND/)
{
#	print $x;<>;
	@temp= split(" ",$x);
	for(my $i=0; $i<scalar(@temp); $i++)
	{
		my @t=split("=",$temp[$i]);
```

```
                        if($t[0] eq "value")
                        {
                                $l=$t[1];
                        }
                }
        }
        if($x =~ m/^\s*<parameter id=UPPER_BOUND/)
        {
                @temp= split(" ",$x);
                for(my $i=0; $i<scalar(@temp); $i++)
                {
                        my @t=split("=",$temp[$i]);
                        if($t[0] eq "value")
                        {
                                $u=$t[1];
                        }
                }
        }
        if($x =~ m/^\s*<parameter id=FLUX_VALUE/)
        {
                @temp= split(" ",$x);
                for(my $i=0; $i<scalar(@temp); $i++)
                {
                        my @t=split("=",$temp[$i]);
                        if($t[0] eq "value")
                        {
                                $fv=$t[1];
                        }
                }
        }
}
my ($cnt1,$cnt2);
if($x =~ m/^\s*<\/reaction>/)
{
        #Adding values to structure for reactions
        $rxn[$c] = reaction->new( id=>$a8, name=>$b8, rev=>$c8, r=>\@ab,
s_r=>\@ab2, p=>\@ab3, s_p=>\@ab4, flux_val=>$fv, ub=>$u, lb=>$l);
        $flag=0;
        my $aaa=$c+1;#reaction counter
        print "Reaction # - \t",$aaa,"\n";
        print "Rection Id - \t",$rxn[$c]->id,"\n","Reaction name -\t",$rxn[$c]-
>name,"\n";
        print f5 $rxn[$c]->id."\n";
        print "Reversible - \t".$rxn[$c]->rev."\n";
        print "Upperbound - \t".$rxn[$c]->ub."\n";
        print f7 "v(".$rxn[$c]->id.") < ".$rxn[$c]->ub."\n";
        print "Lowerbound - \t".$rxn[$c]->lb."\n";
        print f7 "v(".$rxn[$c]->id.") > ".$rxn[$c]->lb."\n";
        print "Flux Value - \t".$rxn[$c]->flux_val."\n";
        if($rxn[$c]->flux_val<0)
        {
                print f6 " ".$rxn[$c]->flux_val." v(".$rxn[$c]->id.")";
        }
        elsif($rxn[$c]->flux_val==0)
        {
```

```
}
else
{
	print f6 " +".$rxn[$c]->flux_val." v(".$rxn[$c]->id.")";
}
print "Reaction - \t";
$cnt1=0;
my $len=scalar(@{$rxn[$c]->r});
my $templen=0;
foreach(@{$rxn[$c]->r})
{
	print ${$rxn[$c]->s_r}[$cnt1]." ".${$rxn[$c]->r}[$cnt1];
	print f2 ${$rxn[$c]->s_r}[$cnt1]." ".${$rxn[$c]->r}[$cnt1];
	$templen++;
	if($templen != $len)
	{
		print"+";
		print f2 " + ";
	}
	$cnt1++;
}
print " --> ";
print f2 " --> ";
$cnt2=0;
$len=scalar(@{$rxn[$c]->p});
$templen=0;
foreach(@{$rxn[$c]->p})
{
	print ${$rxn[$c]->s_p}[$cnt2]." ".${$rxn[$c]->p}[$cnt2];
	print f2 " ".${$rxn[$c]->s_p}[$cnt2]." ".${$rxn[$c]->p}[$cnt2];
	$templen++;
	if($templen != $len)
	{
		print " + ";
		print f2 " + ";
	}
	$cnt2++;
}
print"\n\n\n";
print f2 "\n";
		print "Reaction - \t";
$cnt1=0;
$len=scalar(@{$rxn[$c]->r});
$templen=0;
foreach(@{$rxn[$c]->r})
{
	print ${$rxn[$c]->s_r}[$cnt1]." ".$s_id{${$rxn[$c]->r}[$cnt1]};
	$templen++;
	if($templen != $len)
	{
		print"+";
	}
	$cnt1++;
}
print " --> ";
```

```
		$cnt2=0;
		$len=scalar(@{$rxn[$c]->p});
		$templen=0;
		foreach(@{$rxn[$c]->p})
		{
			print ${$rxn[$c]->s_p}[$cnt2]." ".$s_id{${$rxn[$c]->p}[$cnt2]};
			$templen++;
			if($templen != $len)
			{
				print " + ";
			}
			$cnt2++;
		}
		print"\n\n\n";
		$c++;
	}
}
exit;
```

Appendix - II

Perl Script for creating stoichiometry matrix:

```
#Generating Stoichiometry matrix, Authors- Shubhangi Kaushik, Devesh Thapar Version-
1.1 Date of Creation- 21-10-2016
#!

use strict;
use warnings;
use Class::Struct;
use Data::Dumper;

open(f2,"reactions.txt"); #File containing reactions
open(f3,"speciesid.txt"); #File containing species ids
open(f1,">","stoichiometric.txt"); #File in which stoichiomatric matrix will be saved
my @rxn;

my @sid;
my $j=0;
foreach (<f3>)
{
	chomp($_);
	$sid[$j]=$_; #species ids added to @sid
	$j++;
}
my @mat;

$j=0;
foreach (<f2>) #Reads reactions file
{
	chomp($_);
	$rxn[$j]=$_;
	my @temp= split("-->",$_);
	my @temp1= split(" ",$temp[0]); #reactants in @temp1
	my @temp2= split(" ",$temp[1]); #products in @temp2
	for(my $i=0; $i<scalar(@temp1); $i++)
	{
		for(my $k=0; $k<scalar(@sid); $k++)
		{
			if( $temp1[$i] =~ m/$sid[$k]/)
			{
				my $val=$i-1;
				$mat[$k][$j]=$temp1[$val]*(-1); #Stoichiometry coefficients
of reactants added to the matrix
			}
		}
	}
	for(my $i=0; $i<scalar(@temp2); $i++)
	{
		for(my $k=0; $k<scalar(@sid); $k++)
		{
			if( $temp2[$i] =~ m/$sid[$k]/)
			{
```

```
                        my $val=$i-1;
                        $mat[$k][$j]=$temp2[$val]; #Stoichiometry coefficients of
products added to the matrix
                    }
                }
        }
        $j++;
}



for(my $y=0; $y<scalar(@sid); $y++) #printing the matrix
{
        for(my $x=0; $x<$j; $x++)
        {

                if(defined $mat[$y][$x])
                {
                        print f1 $mat[$y][$x]."\t";
                }
                else
                {
                        $mat[$y][$x]=0;
                        print f1 $mat[$y][$x]."\t";
                }
        }
        print f1"\n";
}
```

Appendix - III

Perl Script for generating Linear Programming input:

```
#Generating LPP, Authors- Shubhangi Kaushik, Devesh Thapar Version- 1.0 Date of
Creation-02-02-2017
#!

use strict;
use warnings;
use Class::Struct;
use Data::Dumper;

open(f1,"reactionid.txt");
open(f2,"stoichiometric.txt");
open(f3,"bounds.txt");
open(f5,"objective_function.txt");
open (f4,">","obj.txt");
my (@r_id,$j);
$j=0;
while (<f1>)#Retrieving reaction ids in array @r_id
{
	chomp($_);
	$r_id[$j]=$_;
	$j++;
}

print f4 "\\* Objective function *\\ \n";
while (<f5>)#Retrieving reaction ids in array @r_id
{
	print f4 "obj: ".$_;
}

my $count=0;
print f4 "\n\n\\* Constraints *\\ \nSubject To\n";
while (<f2>)#S.v=0 to obtain constraints
{
	chomp($_);
	$count++;
	print f4 "one_$count: ";
	my @temp= split("\t",$_);
	for(my $i=0 ; $i<scalar(@temp); $i++)
	{
		if($temp[$i] lt 0)
		{
			print f4 " ".$temp[$i]." v(".$r_id[$i].")";
		}
		elsif($temp[$i] eq 0)
		{
		}
		else
		{
```

```
                print f4 " +".$temp[$i]." v(".$r_id[$i].")";
            }
        }
        print f4 " = 0\n";
}

print f4"\n\n\\* Variable bounds *\\ \nBounds\n";
while (<f3>)
{
        print f4 $_;
}
```

References

[1] K. Raman and N. Chandra, "Flux balance analysis of biological systems: applications and challenges", *Briefings in Bioinformatics*, vol. 10, no. 4, pp. 435-449, 2009.

[2] J. Orth, I. Thiele and B. Palsson, "What is flux balance analysis?", *Nature Biotechnology*, vol. 28, no. 3, pp. 245-248, 2010.

[3] S. Becker, A. Feist, M. Mo, G. Hannum, B. Palsson and M. Herrgard, "Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox", *Nature Protocols*, vol. 2, no. 3, pp. 727-738, 2007.

[4] M. Kanehisa, "KEGG: Kyoto Encyclopedia of Genes and Genomes", *Nucleic Acids Research*, vol. 28, no. 1, pp. 27-30, 2000.

[5] P. Karp, "Expansion of the BioCyc collection of pathway/genome databases to 160 genomes", *Nucleic Acids Research*, vol. 33, no. 19, pp. 6083-6089, 2005.

[6] Joshi-Tope, G., et al. "Reactome: a knowledgebase of biological pathways", *Nucleic acids research* 33.suppl 1 (2005): D428-D432.

[7] I. Schomburg, "BRENDA, enzyme data and metabolic information", *Nucleic Acids Research*, vol. 30, no. 1, pp. 47-49, 2002.

[8] K. Kauffman, P. Prakash and J. Edwards, "Advances in flux balance analysis", *Current Opinion in Biotechnology*, vol. 14, no. 5, pp. 491-496, 2003.

[9] K. Raman, P. Rajagopalan and N. Chandra, "Flux Balance Analysis of Mycolic Acid Pathway: Targets for Anti-Tubercular Drugs", *PLoS Computational Biology*, vol. 1, no. 5, p. e46, 2005.

[10] J. Edwards and B. Palsson, "The Escherichia coli MG1655 in silico metabolic genotype: Its definition, characteristics, and capabilities", *Proceedings of the National Academy of Sciences*, vol. 97, no. 10, pp. 5528-5533, 2000.

[11] MATLAB - The Language Of Technical Computing. http://www.mathworks.com/products/matlab/[The MathWorks, Inc]

[12] J. Wright and A. Wagner, "The Systems Biology Research Tool: evolvable open-source software", *BMC Systems Biology*, vol. 2, no. 1, p. 55, 2008.

[13] I. Rocha, P. Maia, P. Evangelista, P. Vilaça, S. Soares, J. Pinto, J. Nielsen, K. Patil, E. Ferreira and M. Rocha, "OptFlux: an open-source software platform for in silico metabolic engineering", *BMC Systems Biology*, vol. 4, no. 1, p. 45, 2010.

[14] T. Garvey, P. Lincoln, C. Pedersen, D. Martin and M. Johnson, "BioSPICE: Access to the Most Current Computational Tools for Biologists", *OMICS: A Journal of Integrative Biology*, vol. 7, no. 4, pp. 411-420, 2003.

[15] K. Raman and N. Chandra, "PathwayAnalyser: A Systems Biology Tool for Flux Analysis of Metabolic Pathways", *Nature Proceedings*, 2008.

[16] Balaji, S., and M. Sundararajan Murugaiyan. "Waterfall vs. V-Model vs. Agile: A comparative study on SDLC." *International Journal of Information Technology and Business Management* 2.1 (2012): 26-30.

[17] Dvorski, Dalibor D. "Installing, configuring, and developing with Xampp." *Skills Canada* (2007).

[18] Vaughan-Nichols, Steven J. "Will HTML 5 restandardize the web?" *Computer* 43.4 (2010): 13-15.

[19] Makhorin, Andrew. "GLPK (GNU Linear Programming Kit), 2000." B B (2014).

[20] D. Lee, H. Yun, S. Park and S. Lee, "MetaFluxNet: the management of metabolic reaction information and quantitative metabolic flux analysis", *Bioinformatics*, vol. 19, no. 16, pp. 2144-2146, 2003.

[21] M. Garcia-Albornoz, S. Thankaswamy-Kosalai, A. Nilsson, L. Varemo, I. Nookaew and J. Nielsen, "BioMet Toolbox 2.0: genome-wide analysis of metabolism and omics data", *Nucleic Acids Research*, vol. 42, no. 1, pp. W175-W181, 2014.

[22] A. Gevorgyan, M. Bushell, C. Avignone-Rossa and A. Kierzek, "SurreyFBA: a command line tool and graphics user interface for constraint-based modeling of genome-scale metabolic reaction networks", *Bioinformatics*, vol. 27, no. 3, pp. 433-434, 2010.

[23] A. Hoppe, S. Hoffmann, A. Gerasch, C. Gille and H. Holzhütter, "FASIMU: flexible software for flux-balance computation series in large metabolic networks", *BMC Bioinformatics*, vol. 12, no. 1, p. 28, 2011.

[24] CPLEX, ILOG. "High-performance software for mathematical programming and optimization." (2005).

[25] Berkelaar, Michel. "lp solve: a Mixed Integer Linear Program solver." (1997).

[26] S. Klamt and A. von Kamp, "An application programming interface for CellNetAnalyzer", *Biosystems*, vol. 105, no. 2, pp. 162-168, 2011.