Similarity calculation of Metabolic Pathways using graph based Algorithms

Submitted to Orissa University of Agriculture and Technology in partial fulfillment of the requirements for the degree of Master of Science in Bioinformatics by Miss Losiana Nayak.

Department of Bioinformatics
Center for Post Graduate studies
OUAT
Unit-3, Bhubaneswar
Orissa.

Dedicated to my
Teachers, family and friends.

CERTIFICATE FROM GUIDE

INDIAN STATISTICAL INSTITUTE

Telephone : (+91) (33) 2575-3105/3100
Telegram : STATISTICA, CALCUTTA 700 108
FAX : (+91) (33) 2578-3357
(+91) (33) 2577-3035/6680
E-mail : rajat@isical.ac.in

203 BARRACKPORE TRUNK ROAD
KOLKATA 700 108, INDIA

Dr. Rajat K. De
Assistant Professor
Machine Intelligence Unit

Date: October 07, 2004.

To Whom It May Concern

This is to certify that **Ms. Losiana Nayak**, a student in the final year of *Master of Science* in *Bioinformatics* of Orissa University of Agriculture and Technology, Bhubaneswar, India, has successfully completed the dissertation titled "Derivation of Graphs from Metabolic Pathways: Similarity Computation", under my supervision. She has involved in this dissertation work during May 10–October 07, 2004. She has been found to be hard working, sincere and efficient.

I wish her success.

Rajat Kumar De

(Rajat Kumar De)

Assistant Professor
MACHINE INTELLIGENCE UNIT
INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road,
Kolkata-700 108.

"**This is to certify that the thesis entitled "Similarity calculation of Metabolic Pathways using graph based Algorithms" submitted by Miss Losiana Nayak to the Orissa University of Agriculture and Technology, Bhubaneswar, in partial fulfillment of the requirements for the degree of Masters of Science in Bioinformatics, has been approved/disapproved by students' advisory committee after an oral examination of the same in collaboration with the external examiner."**

Advisory Committee:

Chairman:	Dr. Rajat K. De Assistant Professor Machine Intelligence Unit Indian Statistical Institute 203 B.T. Road, Kolkata-700108 West Bengal, India rajat@isical.ac.in
Internal Advisor:	Dr. Pratima Roy HOD, Dept. of Bioinformatics Center for Post Graduate Studies Orissa University of Agriculture and Technology Bhubaneswar, Orissa, India
External Examiner:	Sunil Kumar Information Officer Institute of Life Science Bhubaneswar, Orissa, India

ACKNOWLEDGEMENT

A project work is a product of acquired knowledge, hard labor and co-ordination. Experience of a project work helps lifelong, in shaping a person's career, be it in research area or corporate sector.

I want to use this opportunity to convey my hearty thanks to Dr R. K. Mishra (Ex-HOD, OUAT), Dr Pratima Roy (present HOD, OUAT), Ms. K. Karunasree (Assistant Professor, OUAT) for easing the procedures to get this project in ISI.

I am facing lack of words to show my feelings for Dr. Rajat. K. De, Assistant Professor, Indian Statistical Institute, Kolkata, my down to earth, extra cool guide, for his encouragement, constant support, co-operation, suggestions and help at every step of this project.

Special thanks go to my local guardian Mr. Surjit Sarkhel, hostel superintendent Mr. Shyaml Biswas and Miss Sanghamitra Basu for their help regarding my lodging.

A lot of thanks to my hostel mates, project mates and friends for making the work atmosphere cordial and jolly.

Lastly I am grateful to my dearest Dorey mama, my unknown special friend and God for their moral and emotional support to overcome bad patches during this six months period.

CONTENTS

X. Output values of similarity matrix (Iteration wise)

10. Operating system
11. Language specification
12. Software specification
13. Conclusion
14. Bibliography

LIST OF TABLES:

LIST OF FIGURES:

USED ABBREVIATIONS:

1. KEGG- Kyoto Encyclopedia of Genes and Genomes
2. EC- Enzyme Commission

ABSTRACT

Metabolic pathway information can be studied and categorized from evolutionary point of view. This purpose is achieved by inter-genus, intra-genus as well as inter-species and intra-species comparison of a single metabolic pathway by pre-designed efficient algorithms, converting the comparison scores into a particular format for phylogenetic analysis. Finally the phylograms obtained, can be compared to those of a well-accepted method of phylogenetic analysis for percentage accuracy.

The metabolic pathway database, Kyoto Encyclopedia of Genes and Genomes (KEGG) is the knowledge base for higher order functional information relying on systematic analysis of gene functions. It provides metabolic pathway information for each completely sequenced organism in xml file format. Each file contains organism-specific metabolic pathway information. We are using these xml files as our input source.

Here we are taking the Citric Acid Cycle of seven species belonging to group Bacteriae. Six belong to Proteobacteria gamma and the rest one to Proteobacteria delta/epsilon category. The metabolic maps are converted to enzyme graphs using certain rules. This is the first step we are bringing some change, as rules proposed by Heymans and Singh to build enzyme graphs were not updated as well as not clearly mentioned. For computation each graph is represented by its adjacency matrix.

The similarity scores were calculated from each pair of metabolic pathways belonging to two different species by using pre-existing Algorithm (Heymans and Singh, 2002). These similarity scores can be converted to distance scores by equation “Distance = 1 – Similarity”. The Distance matrix constructed from a set of distance scores can be used to build phylogenetic tree using any convenient phylogenetic analysis package that can convert distance scores into a tree for further analysis and comparison study purpose.

INTRODUCTION

Metabolic Pathways

Definition:

A pathway is a linked set of reactions.

A ⟶ C ⟶ E

"In biochemistry, a metabolic pathway is a series of chemical reactions within a biological cell, catalyzed by enzymes, which either results in the removal of a molecule from the environment to be used/stored by the cell (metabolic sink), or the initiation of another metabolic pathway (also called flux generating step)."

Most metabolic pathways have common properties:

- They are irreversible, usually because the first step is a committed step that only runs in one direction.
- The pathways are regulated, usually by feedback inhibition.
- Anabolic and catabolic pathways in eukaryotes are separated by either compartmentation or by the use of different enzymes and cofactors.

Anabolic reactions

- Use small molecules to build large ones
- Require energy

Catabolic reactions

- Break down large molecules
- Provide energy as ATP

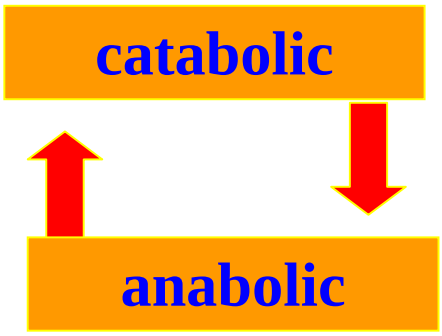


Diagram 1

Relation: Catabolism and Anabolism

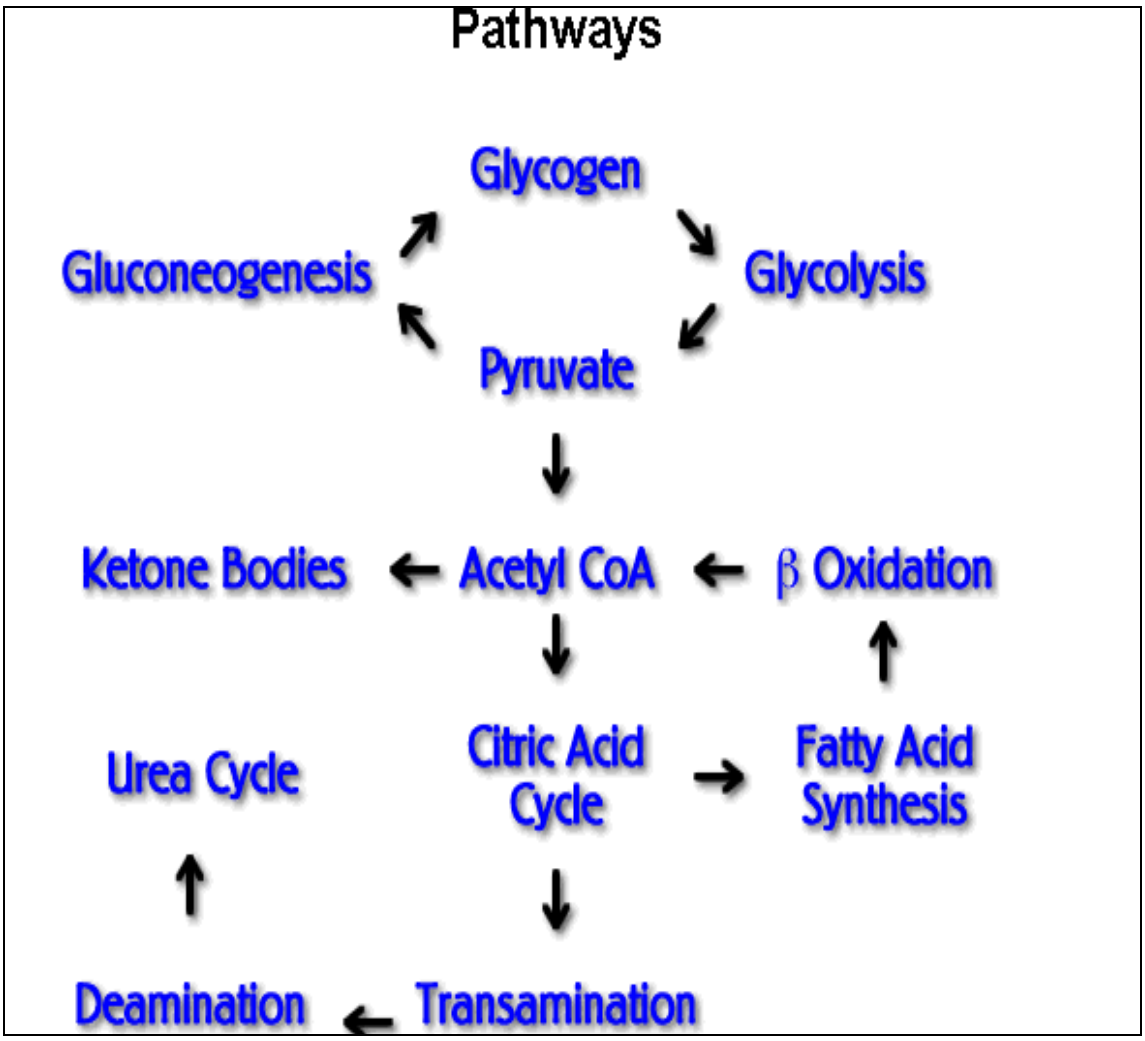
Major Metabolic Pathways

- Krebs cycle (Tricarboxylic acid cycle)

- Pentose phosphate pathway (Hexose monophosphate shunt)
- Glycolysis
- Gluconeogenesis
- Urea cycle
- Fatty acid oxidation (Beta oxidation)

Diagram 2:

Interrelationship of metabolic pathways



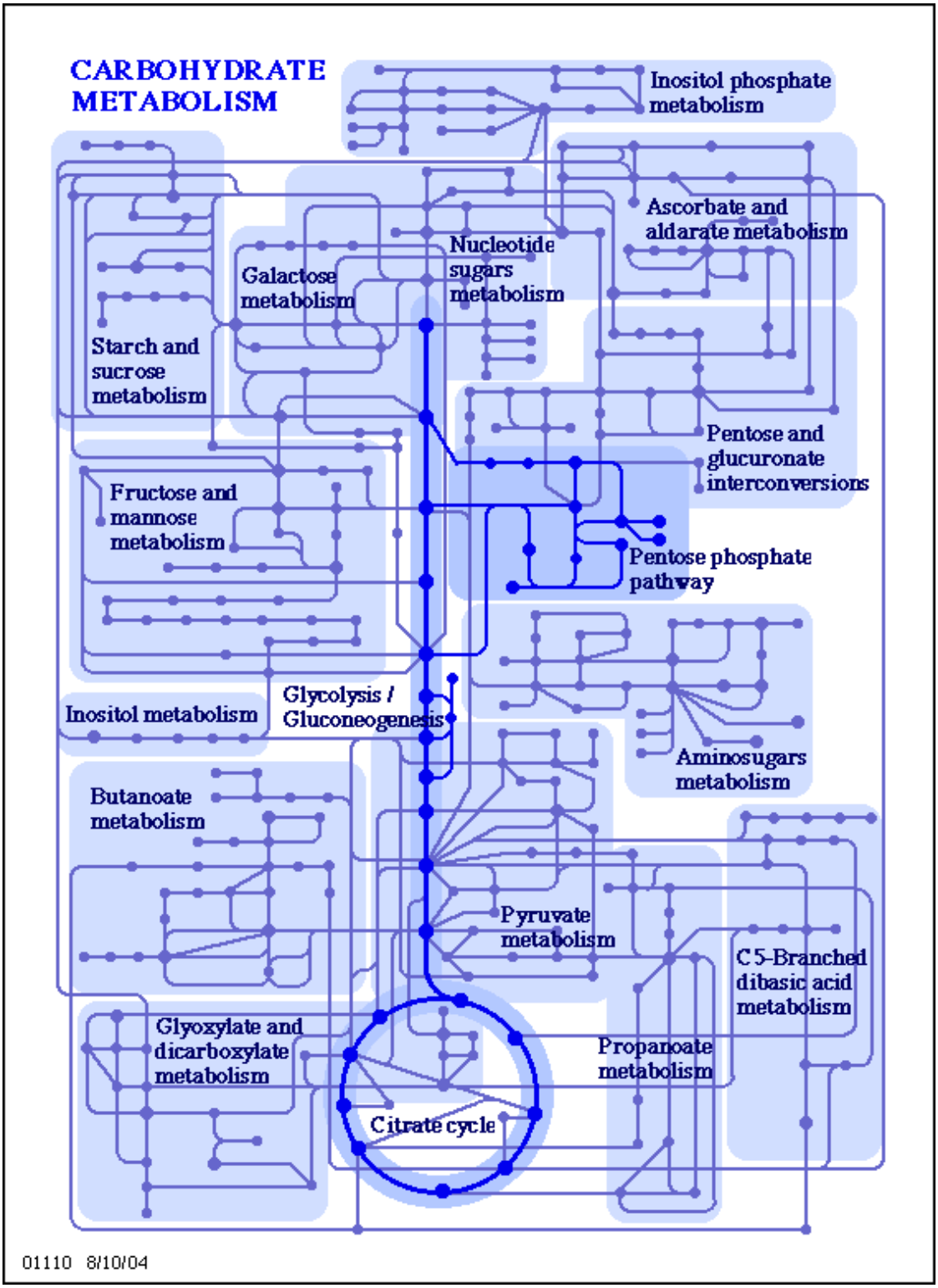
Carbohydrates occupy a central role in catabolism.

1. **Glycolysis** converts sugars to 3C unit (pyruvate) to produce ATP and NADH. Pyruvate can be further utilized in anaerobic (fermentation) or aerobic processes to give CO2 and acetyl CoA.

2. **The Citric acid cycle** converts acetyl CoA to CO_2 and reducing equivalents (NADH and FADH2).

3. **Oxidative Phosphorylation** generates ATP while using electrons from O_2 to regenerate the electron acceptors NAD^+ and FAD.

Diagram 3
Carbohydrate Metabolism


CARBOHYDRATE METABOLISM
Inositol phosphate metabolism
Ascorbate and aldarate metabolism
Nucleotide sugars metabolism
Galactose metabolism
Starch and sucrose metabolism
Pentose and glucuronate interconversions
Fructose and mannose metabolism
Pentose phosphate pathway
Inositol metabolism
Glycolysis / Gluconeogenesis
Aminosugars metabolism
Butanoate metabolism
Pyruvate metabolism
C5-Branched dibasic acid metabolism
Glyoxylate and dicarboxylate metabolism
Propanoate metabolism
Citrate cycle
01110 8/10/04

CITRIC ACID CYCLE:

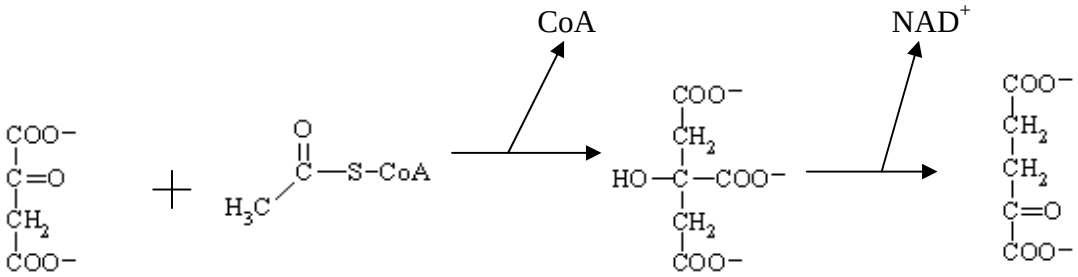
A reaction series that

- Operates under aerobic conditions only
- Oxidizes the 2 carbon atoms of acetyl CoA to CO_2
- Provides reduced coenzymes

CH_3–C –CoA $\Longrightarrow$ 2 CO_2 , $FADH_2$, 3 NADH, + ATP

Acetyl CoA

Steps 1-3 in Citric Acid Cycle

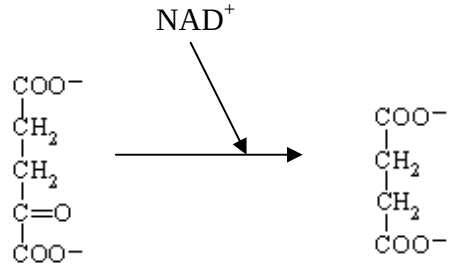


oxaloacetate acetyl CoA citrate β-ketoglutarate

+ CO_2 + NADH

Steps 4-5 of citric acid cycle

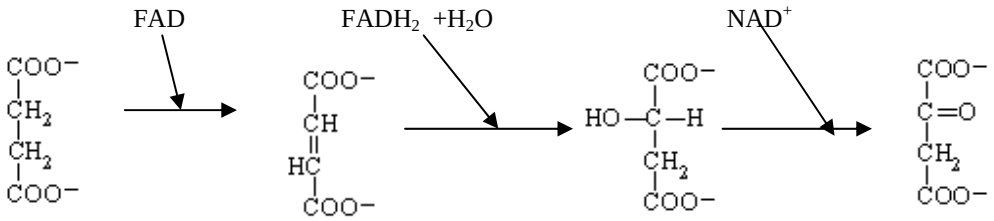
In the next reactions, α-ketoglutarate is oxidized to succinate.



α-ketoglutarate succinate + CO_2 + NADH

Steps 6-8 of citric acid cycle

More oxidations convert succinate to oxaloacetate. The C=C requires FAD



succinate fumerate malate oxaloacetate

+ NADH

One Turn of the Cycle

Acetyl-CoA + 3 NAD^+ + FAD + GDP + P_i + 3 H_2O - - -> 3 NADH + $FADH_2$ + CoA-SH + GTP + 3 CO_2

ATP Energy from Citric Acid Cycle

One turn of the citric acid cycle

3 NADH	x 3 ATP	=	9 ATP
1 $FADH_2$	x 2 ATP	=	2 ATP
1 GTP	x 1 ATP	=	1 ATP
	Total	=	12 ATP

Glucose provides two acetyl COA molecules for two turns of citric acid cycle

2 acetyl CoA ⟶ 24 ATP + 4 CO_2

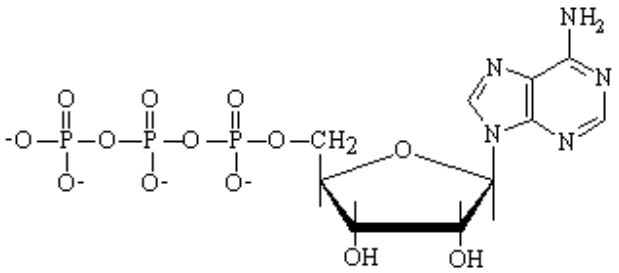


Diagram 4

Structure of ATP

Substrate level regulation of the citric acid cycle

The citric acid cycle is regulated mostly by substrate availability, product inhibition and by some cycle intermediates.

- **pyruvate dehydrogenase:** is inhibited by its products, acetyl-CoA and NADH
- **citrate synthase**: is inhibited by its product, citrate. It is also inhibited by NADH and succinyl-CoA (which signal the abundance of citric acid cycle intermediates).
- **isocitrate dehydrogenase and α-ketoglutarate dehydrogenase**: like citrate synthase, these are inhibited by NADH and succinyl-CoA. Isocitrate dehydrogenase is also inhibited by ATP and stimulated by ADP.

All aforementioned dehydrogenases are stimulated by Ca^{2+}. This makes sense in the muscle, since Ca^{2+} release from the sarcoplasmic reticulum triggers muscle contraction, which requires a lot of energy. This way, the same "second messenger" activates an energy-demanding task and the means to produce that energy.

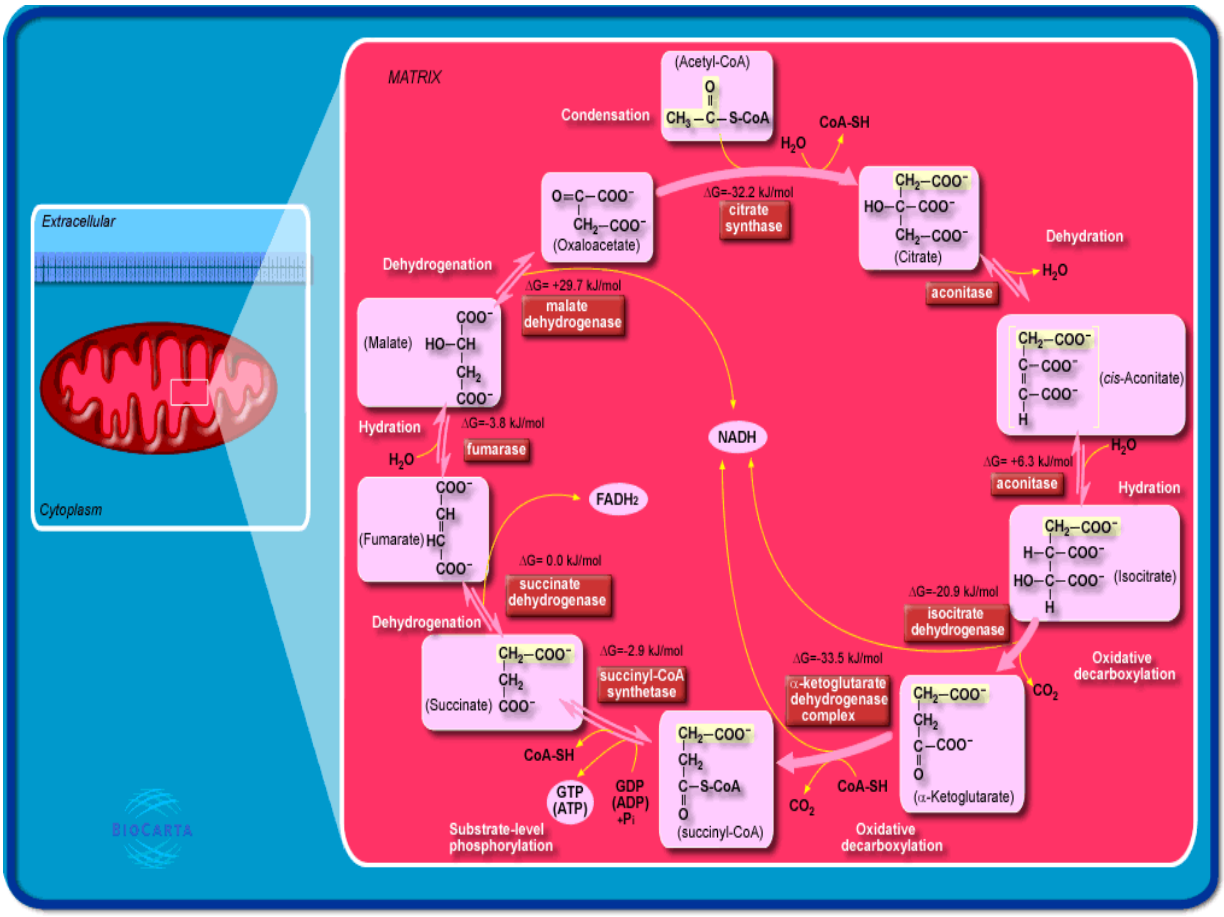
Product/Coenzyme level regulation of Citric Acid Cycle

- Operates when ATP is needed
- High levels of ATP and/or NADH inhibit citrate synthetase (first step in cycle)

- High levels of ADP and NAD^+ activate isocitrate dehydrogenase
- Low levels of ATP or high levels of acetyl CoA speed up the cycle to give energy

Diagram 5

The TCA Cycle



PATHWAY DATABASES:

Biocatalysis/Biodegradation Database

[University of Minnesota] "This database contains microbial biocatalytic reactions and biodegradation pathways primarily for the xenobiotic and the chemical compounds."

Biochemical Pathways

[ExPASy] A searchable index of biochemical pathways

http://www.expasy.ch/cgi-bin/search-biochem-index

Biocyc

[SRI International] The BioCyc Knowledge Library is the new name for EcoCyc and MetaCyc. It is a collection of Pathway/Genome Databases. Each database in the BioCyc collection describes the genome and metabolic pathways of a single organism, with the exception of the MetaCyc database, which is a reference source on metabolic pathways from many organisms. (Refer Table 1)

http://biocyc.org/

Biomolecular Interaction Network

The page describes BIND, which will span the complexity of interaction information gathered through experimental studies of biomolecular interactions. BIND contains interaction, molecular complex, and pathway record.

http://bioinfo.mshri.on.ca/cgi-bin/bind/dataman

Cell Signaling Networks Database

[National Institute of Health Sciences, Japan] CSNDB "is a data- and knowledge-base for signaling pathways of human cells. It compiles the information on biological molecules, sequences, structures, functions, and biological reaction which transfer the cellular signals."

http://geo.nihs.go.jp/csndb/

Enzymology Database

[Argonne National Laboratories] The EMP database contains detailed information on a very large number of enzymes from over 17,000 articles. It can be searched by name, EC number, taxon, tissue or organelle.

http://wit.mcs.anl.gov/EMP/

Table1

Databases under BioCyc

Literature derived Pathway/Genome Databases	
EcoCyc	-- *Escherichia coli* K12
MetaCyc	-- Metabolic pathways and enzymes from 150 species
Computationally-derived Pathway/Genome Databases	
AgroCyc	-- *Agrobacterium tumefaciens*
AnthraCyc	-- *Bacillus anthracis*
BsubCyc	-- *Bacillus subtilis*
CtraCyc	-- *Chlamydia trachomatis*
CauloCyc	-- *Caulobacter crescentus*
EcoO157Cyc	-- *Escherichia coli* O157:H7
HinCyc	-- *Haemophilus influenzae*
HpyCyc	-- *Helicobacter pylori*
HumanCyc	-- *Homo sapiens*
MtbCdcCyc	-- *Mycobacterium tuberculosis* CDC1551
MtbRvCyc	-- *Mycobacterium tuberculosis* H37Rv
MpneuCyc	-- *Mycoplasma pneumoniae*
PlasmoCyc	-- *Plasmodium falciparum*
ShigellaCyc	-- *Shigella flexneri*
TpalCyc	-- *Treponema pallidum*
VchoCyc	-- *Vibrio cholerae*
Other Pathway/Genome Databases on the Internet	
Institution or Project	**Database**
TAIR	-- *Arabidopsis thaliana*
EBI	-- *Methanococcus jannaschii*
Pseudomonas Genome Project	-- *Pseudomonas aeruginosa*
SGD	-- *Saccharomyces cerevisiae*
Carnegie Inst.	-- *Synechocystis* PCC6803

Kyoto Encyclopedia of Genes and Genomes

[GenomeNet] "KEGG is an effort to computerize current knowledge of molecular and cellular biology in terms of the information pathways that consist of interacting molecules or genes and to provide links from the gene catalogs produced by genome sequencing projects."

http://www.genome.ad.jp/kegg/kegg.html

Links to Pathway and Other Databases

[GenomeNet] A comprehensive list of links to biological and biochemical databases. The types of databases include metabolic pathways; enzymes, compounds, and elements; regulatory pathways; protein-protein interactions; nomenclature and classification, taxonomy,complete genomes and analysis. http://www.genome.ad.jp/kegg/kegg4.html

Links to Pathway Databases

[Kyoto University] A list of links to databases of pathways of reactions and compounds in living cells

http://www-mbi3.kuicr.kyoto-u.ac.jp/~kihara/biolink/pathdb.html

A handy description of pathway databases is provided in Table 2.

TABLE 2

Databases of Pathways (for reactions & compounds in living cells)

Name	Contents	Site
KEGG	Pathway maps of metabolic & regulatory pathways	Inst. for Chemical Res., Kyoto Univ. (Japan)
LIGAND	Ligand Chemical Database for Enzyme Reactions	Inst. for Chemical Res., Kyoto Univ. (Japan)
BRITE	Cell cycle pathway maps	Inst. for Chemical Res., Kyoto Univ. (Japan)

SPAD	Signaling pathway maps and proteins of cells	Kyushu Univ. (Japan)
CSNDB	Signaling pathway maps, drugs	National Institute of Health Sciences (Japan)
PUMA	Maps of metabolic pathways, reactions, protein sequences	Argonne National Laboratory (USA)
GenoBase	Maps of metabolic pathways, reactions, protein sequences	National Institute of Health (USA)
EcoCyc	E.coli: Maps of metabolic pathways, reactions, protein sequences	Pangea Systems Inc.(USA)
WIT	Maps of metabolic pathways, reactions, protein sequences	Michigan State Univ. (USA)
Soybase0	Maps of metabolic pathways. The metabolic component of SoyBase, a soybean genetic database.	Yale Univ. (USA)
ENZYME	Enzyme flat DB: Reactions, EC number	Centre Medical Universitaire (Switzerland)
GIF-DB	D. melanogaster Regulation reactions of genes in morphogenesis	Centre Nat. de la Recherche Scientifique (France)
Klotho	Lists of compounds in cells, figures	Washington Univ. (USA)
EC Enzyme DB	Enzyme entry Search from OMIM, Swissprot	Johns Hopkins Univ. (USA)
Chemfinder	Compound Search from name, melting/boiling point, CAS number, MW	CambridgeSoft Co. (USA)

However detailed information can be obtained from links given below.

Links to Pathway and Other Databases:

Metabolic Pathways

- KEGG Metabolic Pathways

- EMP - Enzymes and Metabolic Pathways
- UM-BBD - Microbial Biocatalysis/Biodegradatation
- EcoCyc - *E. coli* Genes and Metabolism
- SoyBase - Soybean Metabolism
- Boehringer Mannheim - Biochemical Pathways
- IUBMB-Nicholson Minimaps
- CFG Glycosylation Pathways

Enzymes, Compounds and Reactions

- LIGAND - Biochemical Compounds and Reactions
- ENZYME - Enzymes
- BRENDA - Comprehensive Enzyme Information System
- Klotho - Biochemical Compounds
- ChemFinder - Searching Chemicals
- ChemIDplus at NLM
- ChemBank - Bioactives Database
- CarbBank - Complex Carbohydrate Structure Database
- LIPIDBANK for Web - Lipids
- Glycoconjugate Data Bank
- PROMISE - Prosthetic Groups and Metal Ions
- WebElements - Periodic Table

Regulatory Pathways

- KEGG Regulatory Pathways
- SPAD - Signal Transduction
- CSNDB - Cell Signaling Networks
- Yeast Pathways in MIPS
- Interactive Fly - *Drosophila* Genes
- GeNet - Gene Networks Database

- HOX-Pro - Homeobox Genes Database
- Wnt Signaling Pathway
- STKE Connection Maps
- AfCS Cellular Signaling Maps

Protein-Protein Interactions

- BRITE Database for Biomolecular Relations
- DIP - Database of Interacting Proteins
- BIND - Biomolecular Interaction Network Database

KEGG: Kyoto Encyclopedia of Genes and Genomes

A grand challenge in the post-genomic era is a complete computer representation of the cell and the organism, which will enable computational prediction of higher-level complexity of cellular processes and organism behaviors from genomic information. A bioinformatics resource named KEGG (initiated in May 1995 under the Human Genome Program of the Ministry of Education, Science, Sports and Culture in Japan) have been developed for this purpose, which is a part of the research projects in the Kanehisa Laboratory of Kyoto University, Bioinformatics Center. KEGG (Kyoto Encyclopedia of Genes and Genomes) is a bioinformatics resource for understanding higher-order functional meanings and utilities of the cell or the organism from its genome information.

KEGG is the primary database resource of the Japanese Genome Net service (http://www.genome.ad.jp/) for understanding higher order functional meanings and utilities of the cell of the organism from its genome information. *KEGG* is a knowledge base for systematic analysis of gene functions, linking genomic information with higher order functional information.

Table 3

The three graph objects in KEGG

Graph	Node	Edge	Main databases
Gene universe	Gene	Any association of genes (ortholog/paralog relation, sequence/structural similarity, adjacency on chromosome, expression similarity)	GENES, SSDB, KO
Chemical universe	Chemical compound (including carbohydrate)	Any association of compounds (chemical reactivity, structural similarity, etc.)	COMPOUND, GLYCAN, REACTION
Protein network	Protein (including other gene products)	Known interaction/relation of proteins (direct protein–protein interaction, gene expression relation, enzyme–enzyme relation)	PATHWAY

KEGG consists of three graph objects for representation and manipulation of genomic, chemical and network data. Mathematically, a graph is a set of nodes (building blocks) and edges (interactions or relations). As shown in Table 3 the three graph objects are called the gene universe (GENES, SSDB and KO databases), the chemical universe (COMPOUND, GLYCAN and REACTION databases) and the protein network (PATHWAY database). The gene universe is a conceptual graph object representing ortholog/ paralog relations, operon information and other relationships between genes in all the completely sequenced genomes. The chemical universe is another conceptual graph object representing chemical reactions and structural/functional relations among metabolites and other biochemical compounds. In contrast, the protein network is based on biological phenomena, representing known molecular interaction networks in various cellular processes.

KEGG databases consist of

- The *PATHWAY* database, for computerized knowledge on molecular interaction networks such as pathways and complexes. Higher order functional information is

stored in it, which contains graphical representations of cellular processes, such as metabolism, membrane transport, signal transduction and cell cycle. It is supplemented by a set of ortholog group tables for the information about conserved sub-pathways (pathway motifs), which are especially useful in predicting gene functions.

- The *GENES* database, a collection of gene catalogs for all the completely sequenced genomes and some partial genomes with up-to-date annotation of gene functions.

- The *LIGAND* database, for information about enzyme molecules and enzymatic reactions that are relevant to cellular processes.

- Limited amounts of experimental data for microarray gene expression profiles and yeast two-hybrid systems in the *EXPRESSION* and *BRITE* databases, respectively.

- A new database named SSDB, for exploring the universe of all protein coding genes in the complete genome and for identifying functional links and ortholog groups.

The data objects in the *KEGG* databases are all represented as graphs and various computational methods are developed to detect graph features that can be related to biological functions. For example the correlated clusters are graph similarities which can be used to predict a set of genes coding for a pathway or a complex, as summerized in the ortholog group tables, and the cliques in the *SSDB* graph are used to annotate genes.

KEGG provides Java graphics tools for browsing genome maps and manipulating expression maps, as well as computational tools for sequence comparison and path computation. The *KEGG* databases are daily updated and freely available at (http://www.genome.ad.jp/kegg/).

PATHWAY Database

(Generalized protein interaction network)

PATHWAY database is a collection of pathway maps, representing wiring diagrams of proteins and other gene products responsible for various cellular functions. Reflecting the map resolution and functional modules at different levels, these pathway maps are hierarchically classified. There are five categories in the top level (metabolism, genetic information processing, environmental information processing, cellular processes and human diseases) and 24 subcategories in the second level. The third level in the hierarchy corresponds to individual pathway maps. When the protein network is linked to the gene universe, the fourth level corresponds to KO (KEGG Orthology) entries. Thus the hierarchy of gene functions in KEGG is based on the hierarchies of the protein network as shown in Table 4.

Graphical diagrams for the reference metabolic pathways represent the Database. Each reference pathway can be viewed as a network of enzymes or a network of Enzyme Commission (EC) numbers. Once enzyme genes were identified in the genome (based on sequence similarity and positional correlation of genes) and the EC numbers were properly assigned, organism specific pathways were constructed computationally by correlating the genes in the genome with gene products (enzymes) in the reference pathways by assigning EC numbers. The EC numbers in the metabolic pathways play roles as identifiers of the nodes (enzymes) and also as keys for linking with the genomic information.

As a whole PATHWAY database contains current knowledge on molecular interaction networks, including metabolic pathways, regulatory pathways and molecular complexes. The data objects stored in the PATHWAY database is called the generalized protein interaction network or simply the network, which is a network of gene products (nodes) with three types of interactions or relations (edges):

Table 4

The hierarchy of KEGG orthology (KO)

01100 Metabolism
01110 Carbohydrate metabolism
01120 Energy metabolism
01130 Lipid metabolism
01140 Nucleotide metabolism
01150 Amino acid metabolism
00251 Glutamate metabolism
.....
00300 Lysine biosynthesis
K00003 E1.1.1.3, thrA; homoserine dehydrogenase
K00928 E2.7.2.4, lysC; aspartate kinase
K00133 E1.2.1.11, asd; aspartate-semialdehyde dehydrogenase
K01714 E4.2.1.52, dapA; dihydrodipicolinate synthase
K00215 E1.3.1.26, dapB; dihydrodipicolinate reductase
K00674 E2.3.1.117, dapD; 2,3,4,5-tetrahydropyridine-2-carboxylate *N*-succinyltransferase
K00821 E2.6.1.17; *N*-succinyldiaminopimelate aminotransferase
K01439 E3.5.1.18, dapE; succinyl-diaminopimelate desuccinylase
K01778 E5.1.1.7, dapF; diaminopimelate epimerase
K01586 E4.1.1.20, lysA; diaminopimelate decarboxylase
.....
00310 Lysine degradation
.....
01160 Metabolism of other amino acids
.....
01200 Genetic information processing
01300 Environmental information processing
01400 Cellular processes
01500 Human disease

- enzyme-enzyme relations, *i.e.*, relation of two enzymes catalyzing successive reaction steps in the metabolic pathway,
- direct protein-protein interactions such as binding and phosphorylation,
- gene expression relations involving transcription factors and target gene products.

The generalized protein interaction is drawn manually as a set of binary relations. The set of binary relations is a computable form of the network information but at the moment only enzyme-enzyme relations are maintained at (http://www.genome.ad.jp/britel/Ecrel/ ecrel.xl) where a relation consists of a pair of nodes (enzymes) and an edge (common compound) in between.

The PATHWAY database is a collection of manually drawn diagrams called the KEGG reference pathway diagrams (maps), each corresponding to a known network of functional significance. From the manually drawn reference pathways, many organism specific pathways are automatically generated according to the ortholog identifier assignments in the GENES database. The organism-specific pathways are demarcated from each other by superimposing (coloring) genes specific to every organism. The total number of gene product nodes that appear on the KEGG pathways is approximately 6,000 and roughly one-quarter to one- third of the genes in a bacterial or archaeal genome that can be mapped to one or more pathway diagrams. The total no of genes in the KEGG ortholog group tables is approximately 26,000, which is ~10% of the total no of genes in the GENES database. As of 25th September 2003, the database contains 13,457 entries including 235 reference pathway diagrams.

In the past, the pathway diagrams were available only in GIF (or PNG) image files. Although the coordinates of nodes (boxes) could be obtained from the HTML file, it was not possible to reconstruct the pathway because the edge information was not readily available. The KEGG Markup Language (KGML) hence been released as a specification of graph objects in KEGG. At present all metabolic pathways and some regulatory pathways are available in KGML, enabling computational reconstruction and manipulation of KEGG pathways to academic users at http://www.genome.ad.jp/kegg/ soap/.

ACCESS METHOD:

The primary mode of access to KEGG is through GenomeNet website at http://www.genome.ad.jp/kegg/. Different components of KEGG resource can most conveniently be accessed from KEGG table of contents page at http://www.genome.ad.jp/kegg/kegg2.html. The four databases for the chemical universe, COMPOUND, GLYCAN, REACTION and ENZYME are collectively called the LIGAND database with a separate home page at http://www.genome.ad.jp/ligand/. Table 5 summarizes these and other useful URLs including additional databases not covered in the present article. For computerized access to KEGG, the SOAP server is open to academic users at http://www.genome.ad.jp/kegg/soap/. All the KEGG databases, except SSDB, are also available to academic users by anonymous FTP at http://www.genome.ad.jp/anonftp/.

Table 5

Useful urls for KEGG Databases

Database/content	URL
KEGG table of contents (PATHWAY, GENES, GENOME, KO, etc.)	http://www.genome.ad.jp/kegg/kegg2.html
SSDB	http://www.genome.ad.jp/kegg/ssdb/
LIGAND (COMPOUND, GLYCAN, REACTION, ENZYME)	http://www.genome.ad.jp/ligand/
EXPRESSION	http://www.genome.ad.jp/kegg/expression/
BRITE	http://www.genome.ad.jp/brite/
KGML	http://www.genome.ad.jp/kegg/xml/
KEGG API	http://www.genome.ad.jp/kegg/soap/
Anonymous FTP	http://www.genome.ad.jp/anonftp/
KEGG home page	http://www.genome.ad.jp/kegg/
DBGET home page	http://www.genome.ad.jp/dbget/
GenomeNet home page	http://www.genome.ad.jp/

ENZYME COMMISSION NUMBERS

I. Introduction

EC numbers **(**Enzyme Commission numbers**)** are a numerical classification scheme for enzymes, based on the chemical reactions they catalyze. As a system of enzyme nomenclature, every EC number is associated with a recommended name for the respective enzyme. Every enzyme code consists of the letters "EC" followed by four numbers separated by periods. Those numbers represent a progressively finer classification of the enzyme.

II. History

The General Assembly of the International Union of Biochemistry (IUB) decided, during the third International Congress of Biochemistry (Brussels) in August 1955, to set up an International Commission on Enzymes. This step was taken in consultation with the International Union of Pure and Applied Chemistry (IUPAC).

The International Commission on Enzymes was established in 1956 by the President of the International Union of Biochemistry, **Professor M. Florkin** with the advice of an *ad hoc* Committee.

The terms of reference of the Enzyme Commission, as laid down by the *ad hoc* Committee, were as follows:

> ***"To consider the classification and nomenclature of enzymes and coenzymes, their units of activity and standard methods of assay, together with the symbols used in the description of enzyme kinetics."***

The Enzyme Commission faced many difficulties arising from the uncontrolled naming of the rapidly increasing number of known enzymes. Some of the names in use were definitely misleading; others conveyed little or nothing about the nature of the reaction catalyzed, as for example, *diaphorase, Zwischenferment, catalase*. Enzymes catalyzing

essentially similar reactions had sometimes names implying that they belong to different groups, while some enzymes of different types had been placed in the same group; for example, the *pyrophosphorylases* had included both glycosyltransferases and phosphotransferases. In some cases a name, which had been well established for many years with a definite meaning, such as the term *synthetase*, had been later employed with different meanings, thus causing confusion.

In 1977 the Nomenclature Committee of the International Union Biochemistry and Molecular Biology (NC-IUB) was set up and responsibility for enzyme nomenclature passed to it. In this task it has worked closely with the IUPAC-IUB Joint Commission on Biochemical Nomenclature (JCBN).

The Enzyme List in the sixth edition contains 3540 entries, of which 178 record enzymes, which are now transferred elsewhere in the list, and 166 have been deleted completely. Thus the number of enzymes actually listed is 3196, an increase of 29% on the 1984 edition.

The size of the list has increased steadily since the publication of the Report of the Enzyme Commission, as shown in the following figures for 'live' entries:

Report of the Enzyme Commission (1961)	712
Enzyme Nomenclature (1964)	875
Enzyme Nomenclature (1972)	1770
Enzyme Nomenclature (1978)	2122
Enzyme Nomenclature (1984)	2477
Enzyme Nomenclature (1992)	3196

III. Rules for Classification and Nomenclature of Enzyme Catalyzed Reactions

In *Enzyme Nomenclature* 1992 there was a section on general principles; recommended and systematic names; scheme of classification and numbering of enzymes; and rules for classification and nomenclature.

1. General principles

Because of their close interdependence, it is convenient to deal with the classification and nomenclature together.

The *first general principle* of these 'Recommendations' is that names purporting to be names of enzymes, especially those ending in *-ase*, should be used only for single enzymes, *i.e.* single catalytic entities. They should not be applied to systems containing more than one enzyme. When it is desired to name such a system on the basis of the overall reaction catalyzed by it, the word *system* should be included in the name. For example, the system catalyzing the oxidation of succinate by molecular oxygen, consisting of succinate dehydrogenase, cytochrome oxidase, and several intermediate carriers, should not be named *succinate oxidase*, but it may be called the *succinate oxidase system*. Other examples of systems consisting of several structurally and functionally linked enzymes (and cofactors) are the *pyruvate dehydrogenase system*, the similar *2-oxoglutarate dehydrogenase system*, and the *fatty acid synthase system*.

In this context it is appropriate to express disapproval of a loose and misleading practice that is found in the biological literature. It consists in designation of a natural substance (or even of an hypothetical active principle), responsible for a physiological or biophysical phenomenon that can’t be described in terms of a definite chemical reaction, by the name of the phenomenon in conjugation with the suffix *-ase*, which implies an individual enzyme. Some examples of such *phenomenase* nomenclature, which should be discouraged even if there are reasons to suppose that the particular agent may have enzymatic properties, are: *permease, translocase, repairase, joinase, replicase, codase, etc.*

The *second general principle* is that enzymes are principally classified and named according to the reaction they catalyze. The chemical reaction catalyzed is the specific property that distinguishes one enzyme from another, and it is logical to use it as the basis for the classification and naming of enzymes.

Several alternative bases for classification and naming had been considered, *e.g.* chemical nature of the enzymes (whether it is a flavoprotein, a hemoprotein, a pyridoxal-phosphate protein, a copper protein, and so on), or chemical nature of the substrate (nucleotides, carbohydrates, proteins, *etc.*). The first cannot serve as a general basis, for only minority of enzymes have such identifiable prosthetic groups. The chemical nature of the enzyme has, however, been used exceptionally in certain cases where classification based on specificity is difficult, for example, with the peptidases (subclass EC 3.4).

Peptidase Nomenclature

The nomenclature of the peptidases is troublesome. Their specificity is commonly difficult to define, depending upon the nature of several amino acid residues around the peptide bond to be hydrolyzed and also on the conformation of the substrate polypeptide chain. A classification involving the additional criterion of catalytic mechanism is therefore used.

It is recommended that the term "peptidase" be used as synonymous with "peptide hydrolase" for any enzyme that hydrolyses peptide bonds. Peptidases are recommended to be further divided into "exopeptidases" that act only near a terminus of a polypeptide chain and "endopeptidases" that act internally in polypeptide chains. The usage of "peptidase" is synonymous with "protease" as it was originally used as a general term for both exopeptidases and endopeptidases, but it should be noted that previously, in *Enzyme Nomenclature (1984)*, "peptidase" was restricted to the enzymes included in sub-subclasses EC 3.4.11-19, the exopeptidases. Also, the term "proteinase" used previously for the enzymes included in sub-subclasses EC 3.4.21-99 carried the same meaning as "endopeptidase", and has been replaced by "endopeptidase" for consistency.

Two sets of sub-subclasses of peptidases are recognized, those of the exopeptidases (EC 3.4.11-19) and those of the endopeptidases (EC 3.4.21-24 and EC 3.4.99). The exopeptidases act only near the ends of polypeptide chains, and those acting at a free N-terminus liberate a single amino-acid residue (aminopeptidases, EC 3.4.11), or a dipeptide or a tripeptide (dipeptidyl-peptidases and tripeptidyl-peptidases, EC 3.4.14). The exopeptidases acting at a free C-terminus liberate a single residue (carboxypeptidases, EC 3.4.16-18) or a dipeptide (peptidyl-dipeptidases, EC 3.4.15). The carboxypeptidases are allocated to four groups on the basis of catalytic mechanism: the serine-type carboxypeptidases (EC 3.4.16), the metallocarboxypeptidases (EC 3.4.17) and the cysteine-type carboxypeptidases (EC 3.4.18). Other exopeptidases are specific for dipeptides (dipeptidases, EC 3.4.13), or remove terminal residues that are substituted, cyclized or linked by isopeptide bonds (peptide linkages other than those of α-carboxyl to α-amino groups) (omega peptidases, 3.4.19).

The endopeptidases are divided into sub-subclasses on the basis of catalytic mechanism, and specificity is used only to identify individual enzymes within the groups. These are the sub-subclasses of serine endopeptidases (EC 3.4.21), cysteine endopeptidases (EC 3.4.22), aspartic endopeptidases (EC 3.4.23), metalloendopeptidases (EC 3.4.24) and threonine endopeptidases (EC 3.4.25). Endopeptidases that could not be assigned to any of the sub-subclasses EC 3.4.21-25 were listed in sub-subclass EC 3.4.99.

In describing the specificity of peptidases, use is made of a model in which the catalytic site is considered to be flanked on one or both sides by specificity subsites, each able to accommodate the side chain of a single amino acid residue. These sites are numbered from the catalytic site, S1...S*n* towards the N-terminus of the substrate, and S1'...S*n'* towards the C-terminus. The residues they accommodate are numbered P1...P*n*, and P1'...P*n'*, respectively, as follows:

```
Substrate: - P3 - P2 - P1 ┼P1'- P2'- P3'-
Enzyme:    - S3 - S2 - S1 * S1'- S2'- S3'-
```

In this representation, the catalytic site of the enzyme is marked *. The peptide bond cleaved (the scissile bond) is indicated by the symbol '+' or a hyphen in the structural formula of the substrate, or a hyphen in the name of the enzyme. In describing the specificity of endopeptidases, the term "oligopeptidase" is used to refer to those that act only on substrates smaller than proteins.

Finally, in describing the specificity of endopeptidases, the term oligopeptidase' is used to refer to those that act optimally on substrates smaller than proteins.

The second basis for classification is hardly practicable, owing to the great variety of substances acted upon and because it is not sufficiently informative unless the type of reaction is also given. It is the overall reaction, as expressed by the formal equation should be taken as the basis. Thus, the intimate mechanism of the reaction and the formation of intermediate complexes of the reactants with the enzyme is not taken into account, but only the observed chemical change produced by the complete enzyme reaction. For example, in those cases in which the enzyme contains a prosthetic group that serves to catalyze transfer from a donor to an acceptor (*e.g.* flavin, biotin, or pyridoxal-phosphate enzymes), the name of the prosthetic group is not normally included in the name of the enzyme. Nevertheless, where alternative names are possible, the mechanism may be taken into account in choosing between them.

A consequence of the adoption of the chemical reaction as the basis for naming enzymes is that a systematic name can’t be given to an enzyme until it is known what chemical reaction it catalyzes. This applies, for example, to a few enzymes that have so far not been shown to catalyze any chemical reaction, but only isotopic exchanges; the isotopic exchange gives some idea of one step in the overall chemical reaction, but the reaction as a whole remains unknown.

A second consequence of this concept is that a certain name designates not a single enzyme protein but a group of proteins with the same catalytic property. Enzymes from different sources (various bacterial, plant or animal species) are classified as one entry.

The same applies to isoenzymes. However, there are exceptions to this general rule. Some are justified because the mechanism of the reaction or the substrate specificity is so different as to warrant different entries in the enzyme list. This applies, for example, to the two cholinesterases, EC 3.1.1.7 and 3.1.1.8, the two citrate hydro-lyases, EC 4.2.1.3 and 4.2.1.4, and the two amine oxidases, EC 1.4.3.4 and 1.4.3.6. Others are mainly historical, *e.g.* acid and alkaline phosphatases (EC 3.1.3.1 and EC 3.1.3.2).

A *third general principle* adopted is that the enzymes are divided into groups on the basis of the type of reaction catalyzed, and this, together with the name(s) of the substrate(s) provides a basis for naming individual enzymes. It is also the basis for classification and code numbers.

Special problems attend the classification and naming of enzymes catalyzing complicated transformations that can be resolved into several sequential or coupled intermediary reactions of different types, all catalyzed by a single enzyme (not an enzyme system). Some of the steps may be spontaneous non-catalytic reactions, while one or more intermediate steps depend on catalysis by the enzyme. Wherever the nature and sequence of intermediary reactions is known or can be presumed with confidence, classification and naming of the enzyme should be based on the first enzyme-catalyzed step that is essential to the subsequent transformations, which can be indicated by a supplementary term in parentheses, *e.g. acetyl-CoA:glyoxylate C-acetyltransferase (thioester-hydrolysing, carboxymethyl-forming)* (EC 2.3.3.9, *cf.* section 3).

To classify an enzyme according to the type of reaction catalyzed, it is occasionally necessary to choose between alternative ways of regarding a given reaction. In general, that alternative should be selected which fits in best with the general system of classification and reduces the number of exceptions.

One important extension of this principle is the question of the direction in which the reaction is written for the purposes of classification. To simplify the classification, the direction chosen should be the same for all enzymes in a given class, even if this

direction has not been demonstrated for all. Thus the *systematic* names, on which the classification and code numbers are based, may be derived from a written reaction, even though only the reverse of this has been actually demonstrated experimentally. In the list the reaction is written to illustrate the classification, *i.e.* in the direction described by the systematic name. However, the *common* name may be based on either direction of reaction, and is often based on the presumed physiological direction.

Many examples of this usage are found. The reaction for EC 1.1.1.9 is written as an oxidation of xylitol by NAD^+, in parallel with all other oxidoreductases in subgroup EC 1.1.1, and the systematic name is accordingly, *xylitol:NAD^+ 2-oxidoreductase (*D-*xylulose-forming)*. However, the common name, based on the reverse direction of reaction, is D-*xylulose reductase*.

2. Common and Systematic Names

The first Enzyme Commission gave much thought to the question of a systematic and logical nomenclature for enzymes and finally recommended that there should be two nomenclatures for enzymes one systematic and one working or trivial. The systematic name of an enzyme, formed in accordance with definite rules, showed the action of an enzyme as exactly as possible, thus identifying the enzyme precisely. The trivial name was sufficiently short for general use, but not necessarily very systematic; in a great many cases it was a name already in current use. The introduction of (often cumbersome) systematic names was strongly criticized. In many cases the reaction catalyzed is not much longer than the systematic name and can serve just as well for identification, especially in conjunction with the code number.

The Commission for Revision of Enzyme Nomenclature discussed this problem at length, and a change in emphasis was made. It was decided to give the trivial names more prominence in the Enzyme List; they now follow immediately after the code number, and are described as Common Name. Also, in the index an asterisk indicates the common

names. Nevertheless, it was decided to retain the systematic names as the basis for classification for the following reasons:

(i) the code number alone is only useful for identification of an enzyme when a copy of the Enzyme List is at hand, whereas the systematic name is self-explanatory;

(ii) the systematic name stresses the type of reaction, the reaction equation does not;

(iii) systematic names can be formed for new enzymes by the discoverer, by application of the rules, but code numbers should **not** be assigned by individuals;

(iv) common names for new enzymes are frequently formed as a condensed version of the systematic name; therefore, the systematic names are helpful in finding common names that are in accordance with the general pattern.

It is recommended that for enzymes that are not the main subject of a paper or abstract, the common names should be used but they should be identified at their first mention by their code numbers and source. Where an enzyme is the main subject of a paper or abstract, its code number, systematic name, or, alternatively, the reaction equation and source should be given at its first mention; thereafter the common name should be used. In the light of the fact that enzyme names and code numbers refer to reactions catalyzed rather than to discrete proteins, it is of special importance to give also the source of the enzyme for full identification; in cases where multiple forms are known to exist, knowledge of this should be included where available.

When a paper deals with an enzyme that is not yet in the Enzyme List, the author may introduce a new name and, if desired, a new systematic name, both formed according to the recommended rules. But a number should be assigned only by the Nomenclature Committee of IUBMB.

The Enzyme List contains one or more references for each enzyme. It should be stressed that no attempt has been made to provide a complete bibliography or to refer to the first description of an enzyme. The references are intended to provide sufficient evidence for

the existence of an enzyme catalyzing the reaction as set out. Where there is a major paper describing the purification and specificity of an enzyme or a major review article, this has been quoted to the exclusion of earlier and later papers. In some cases separate references are given for animal, plant and bacterial enzymes.

3. Scheme of Classification and Numbering of Enzyme Catalyzed Reactions

The first Enzyme Commission, in its report in 1961, devised a system for classification of enzymes that also serves as a basis for assigning code numbers to them. These code numbers, prefixed by EC, which are now widely in use, contain four elements separated by points (“.”), with the following meaning:

(i) the first number shows to which of the six main divisions (classes) the enzyme

belongs,

(ii) the second figure indicates the subclass,

(iii) the third figure gives the sub-subclass,

(iv) the fourth figure is the serial number of the enzyme in its sub-subclass.

The subclasses and sub-subclasses are formed according to principles indicated below.

The main divisions and subclasses are:

Class 1. Oxidoreductases

To this class belong all enzymes catalyzing oxidoreduction reactions. The substrate that is oxidized is regarded as hydrogen donor. The systematic name is based on *donor:acceptor oxidoreductase*. The common name will be *dehydrogenase*, wherever this is possible; as an alternative, *reductase* can be used. *Oxidase* is only used in cases where O_2 is the acceptor.

The second figure in the code number of the oxidoreductases, unless it is 11, 13, 14 or 15, indicates the group in the hydrogen (or electron) donor that undergoes oxidation: 1 denotes a -CHOH- group, 2 a -CHO or -CO-COOH group or carbon monoxide, and so on, as listed in the key.

The third figure, except in subclasses EC 1.11, EC 1.13, EC 1.14 and EC 1.15, indicates the type of acceptor involved: 1 denotes $NAD(P)^+$, 2 a cytochrome, 3 molecular oxygen, 4 a disulfide, 5 a quinone or similar compound, 6 a nitrogenous group, 7 an iron-sulfur protein and 8 a flavin. In subclasses EC 1.13 and EC 1.14 a different classification scheme is used and sub-subclasses are numbered from 11 onwards.

It should be noted that in reactions with a nicotinamide coenzyme this is always regarded as acceptor, even if this direction of the reaction is not readily demonstrated. The only exception is the subclass EC 1.6, in which NAD(P)H is the donor; some other redox catalyst is the acceptor.

Although not used as a criterion for classification, the two hydrogen atoms at carbon-4 of the dihydropyridine ring of nicotinamide nucleotides are not equivalent in that the hydrogen is transferred stereospecifically.

Class 2. Transferases

Transferases are enzymes transferring a group, *e.g.* a methyl group or a glycosyl group, from one compound (generally regarded as donor) to another compound (generally regarded as acceptor). The systematic names are formed according to the scheme *donor:acceptor grouptransferase*. The common names are normally formed according to *acceptor grouptransferase* or *donor grouptransferase*. In many cases, the donor is a cofactor (coenzyme) charged with the group to be transferred. A special case is that of the transaminases.

Some transferase reactions can be viewed in different ways. For example, the enzyme-catalysed reaction

X-Y + Z = X + Z-Y

may be regarded either as a transfer of the group Y from X to Z, or as a breaking of the X-Y bond by the introduction of Z. Where Z represents phosphate or arsenate, the process is often spoken of as 'phosphorolysis' or 'arsenolysis', respectively, and a number of enzyme names based on the pattern of *phosphorylase* have come into use. These names are not suitable for a systematic nomenclature, because there is no reason to single out these particular enzymes from the other transferases, and it is better to regard them simply as *Y-transferases.*

In the above reaction, the group transferred is usually exchanged, at least formally, for hydrogen, so that the equation could more strictly be written as:

X-Y + Z-H = X-H + Z-Y.

Another problem is posed in enzyme-catalyzed transaminations, where the $-NH_2$ group and -H are transferred to a compound containing a carbonyl group in exchange for the =O of that group, according to the general equation:

R^1-CH(-NH_2)-R^2 + R^3-CO-R^4 $\rightarrow$ R^1-CO-R^2 + R^3-CH(-NH_2)-R^4.

The reaction can be considered formally as oxidative deamination of the donor (*e.g.* amino acid) linked with reductive amination of the acceptor (*e.g.* oxo acid), and the transaminating enzymes (pyridoxal-phosphate proteins) might be classified as oxidoreductases. However, the unique distinctive feature of the reaction is the transfer of the amino group (by a well-established mechanism involving covalent substrate-coenzyme intermediates), which justified allocation of these enzymes among the transferases as a special subclass (EC 2.6.1, *transaminases*).

The second figure in the code number of transferases indicates the group transferred; a one-carbon group in EC 2.1, an aldehydic or ketonic group in EC 2.2, an acyl group in EC 2.3 and so on.

The third figure gives further information on the group transferred; *e.g.* subclass EC 2.1 is subdivided into *methyltransferases* (EC 2.1.1), *hydroxymethyl-* and *formyltransferases*

(EC 2.1.2) and so on; only in subclass EC 2.7, does the third figure indicate the nature of the acceptor group.

Class 3. Hydrolases

These enzymes catalyze the hydrolytic cleavage of C-O, C-N, C-C and some other bonds, including phosphoric anhydride bonds. Although the systematic name always includes *hydrolase*, the common name is in many cases formed by the name of the substrate with the suffix *-ase*. It is understood that the name of the substrate with this suffix means a hydrolytic enzyme.

A number of hydrolases acting on ester, glycosyl, peptide, amide or other bonds are known to catalyze not only hydrolytic removal of a particular group from their substrates, but likewise the transfer of this group to suitable acceptor molecules. In principle, all hydrolytic enzymes might be classified as transferases, since hydrolysis itself can be regarded as transfer of a specific group to water as the acceptor. Yet in most cases the reaction with water as the acceptor was discovered earlier and is considered as the main physiological function of the enzyme. This is why such enzymes are classified as hydrolases rather than as transferases.

Some hydrolases (especially some of the esterases and glycosidases) pose problems because they have a very wide specificity and it is not easy to decide if two preparations described by different authors (perhaps from different sources) have the same catalytic properties or if they should be listed under separate entries. An example is *vitamin A esterase* (formerly EC 3.1.1.12, now believed to be identical with EC 3.1.1.1). To some extent the choice must be arbitrary; however, separate entries should be given only when the specificities are sufficiently different.

Another problem is that proteinases have 'esterolytic' action; they usually hydrolyse ester bonds in appropriate substrates even more rapidly than natural peptide bonds. In this case classification among the peptide hydrolases is based on historical priority and presumed physiological function.

The second figure in the code number of the hydrolases indicates the nature of the bond hydrolysed; EC 3.1 are the *esterases*; EC 3.2 the *glycosylases*, and so on.

The third figure normally specifies the nature of the substrate, *e.g.* in the esterases the *carboxylic ester hydrolases* (EC 3.1.1), *thiolester hydrolases* (EC 3.1.2), *phosphoric monoester hydrolases* (EC 3.1.3), in the glycosylases the *O-glycosidases* (EC 3.2.1), *N-glycosylases* (EC 3.2.2), *etc*. Exceptionally, in the case of the peptidyl-peptide hydrolases the third figure is based on the catalytic mechanism as shown by active center studies or the effect of pH.

Class 4. Lyases

Lyases are enzymes cleaving C-C, C-O, C-N, and other bonds by elimination, leaving double bonds or rings or conversely adding groups to double bonds. The systematic name is formed according to the pattern *substrate group-lyase*. The hyphen is an important part of the name, and to avoid confusion should not be omitted, *e.g. hydro-lyase* not 'hydrolyase'. In the common names expressions like *decarboxylase, aldolase, dehydratase* (in case of elimination of CO_2, aldehyde or water) are used. In cases where the reverse reaction is much more important or the only one demonstrated, *synthase* (not synthetase) may be used in the name. Various subclasses of the lyases include pyridoxal-phosphate enzymes that catalyze the elimination of a β- or γ-substituent from an α-amino acid followed by a replacement of this substituent by some other group. In the overall replacement reaction, no unsaturated endproduct is formed; therefore these enzymes might formally be classified as *alkyl-transferases* (EC 2.5.1..). However, there is ample evidence that the replacement is a two-step reaction involving the transient formation of enzyme-bound α,β (or β,γ)-unsaturated amino acids. According to the rule that the first reaction is indicative for classification, these enzymes are correctly classified as *lyases*. Examples are *tryptophan synthase* (EC 4.2.1.20) and *cystathionine β-synthase* (EC 4.2.1.22).

The second figure in the code number indicates the bond broken: EC 4.1 are carbon-carbon lyases, EC 4.2 carbon-oxygen lyases and so on.

The third figure gives further information on the group eliminated (*e.g.* CO_2 in EC 4.1.1, H_2O in EC 4.2.1).

Class 5. Isomerases

These enzymes catalyze geometric or structural changes within one molecule. According to the type of isomerism, they may be called *racemases, epimerases, cis-trans-isomerases, isomerases, tautomerases, mutases* or *cycloisomerases*.

In some cases, the interconversion in the substrate is brought about by an intramolecular oxidoreduction (EC 5.3); since hydrogen donor and acceptor are the same molecule, and no oxidized product appears, they are not classified as oxidoreductases, even though they may contain firmly bound $NAD(P)^+$.

The subclasses are formed according to the type of isomerism, the sub-subclasses to the type of substrates.

Class 6. Ligases.

Ligases are enzymes catalyzing the joining together of two molecules coupled with the hydrolysis of a diphosphate bond in ATP or a similar triphosphate. The systematic names are formed on the system *X-Y ligase (ADP-forming).* In earlier editions of the list the term *synthetase* has been used for the common names. Many authors have been confused by the use of the terms *synthetase* (used only for Group 6) and *synthase* (used throughout the list when it is desired to emphasis the synthetic nature of the reaction). Consequently NC-IUB decided in 1983 to abandon the use of synthetase for common names, and to replace them with names of the type *X-Y ligase*. In a few cases in Group 6, where the reaction is more complex or there is a common name for the product, a synthase name is used (*e.g.* EC 6.3.2.11 and EC 6.3.5.1).

It is recommended that if the term *synthetase* is used by authors, it should continue to be restricted to the ligase group.

The second figure in the code number indicates the bond formed; EC 6.1 for C-O bonds (enzymes acylating tRNA), EC 6.2 for C-S bonds (acyl-CoA derivatives) *etc.* Sub-subclasses are only in use in the C-N ligases.

In a few cases it is necessary to use the word *other* in the description of subclasses and sub-subclasses. They have been provisionally given the figure 99 in order to leave space for new subdivisions.

From time to time some enzymes have been deleted from the list while some others have been renumbered. However the old numbers have not been allotted to new enzymes; rather the place has been left vacant and cross-reference is made according to the following scheme:

[**EC 1.2.3.4 deleted entry:** old name]

or

[**EC 1.2.3.4 transferred entry:** now EC 5.6.7.8 - common name].

Entries for reclassified enzymes transferred from one position in the List to another are followed, for reference by a comment indicating the former number.

It is regarded as important that the same policy be followed in future revisions and extensions of the enzyme list, which may become necessary from time to time.

4. Rules for Classification and Nomenclature

(a) General Rules for Systematic Names and Guidelines for Common Names

Rule 1.

(Common Names)

Generally accepted trivial names of substrates may be used in enzyme names. The prefix D- should be omitted for all D-sugars and L- for individual amino acids, unless ambiguity would be caused. In general, it is not necessary to indicate positions of substituents in common names, unless it is necessary to prevent two different enzymes having the same name. The prefix *keto* is no longer used for derivatives of sugars in which -CHOH- has been replaced by -CO-; they are named throughout as dehydro-sugars.

(Systematic Names)

To produce usable systematic names, accepted trivial names of substrates forming part of the enzyme names should be used. Where no accepted and convenient trivial names exist, the official IUPAC rules of nomenclature should be applied to the substrate name. The 1,2,3 system of locating substituents should be used instead of the α,β,γ system, although group names such as β-aspartyl-, γ-glutamyl-, and also β-alanine and γ-lactone are permissible; α,β should normally be used for indicating configuration, as in α-D-glucose. For nucleotide groups, *adenylyl* (not adenyl), *etc*. should be the form used. The name oxo acids (not keto acids), may be used as a class name and for individual compounds in which $-CH_2-$ has been replaced by -CO-, oxo should be used.

Rule 2.

Where the substrate is normally in the form of an anion, its name should end in *-ate* rather than *-ic; e.g. lactate dehydrogenase,* not 'lactic dehydrogenase' or 'lactic acid dehydrogenase'.

Rule 3.

Commonly used abbreviations for substrates *e.g.* ATP, may be used in names of enzymes but the use of new abbreviations (not listed in recommendations of the IUPAC-IUB Commission on Biochemical Nomenclature) should be discouraged. Chemical formulae should not normally be used instead of names of substrates. Abbreviations for names of enzymes *e.g.* GDH, should not be used.

Rule 4.

Names of substrates composed of two nouns such as glucose phosphate, which are normally written with a space should be hyphenated when they form part of the enzyme names and thus become adjectives, *e.g. glucose-6-phosphate 1-dehydrogenase* (EC 1.1.1.49).

Rule 5.

The use as enzyme names of descriptions such as *condensing enzyme, acetate-activating enzyme, pH 5 enzyme* should be discontinued as soon as the catalyzed reaction is known. The word *activating* should not be used in the sense of converting the substrate into a substance that reacts further; all enzymes act by activating their substrates and the use of the word in this sense may lead to confusion.

Rule 6.

(Common Names)

If it can be avoided, a common name should not be based on a substance that is not a true substrate *e.g.* enzyme EC 4.2.1.17 should not be called 'crotonase', since it does not act on crotonate.

Rule 7.

(Common Names)

Where a name in common use, gives some indication of the reaction and is not incorrect or ambiguous its continued use is recommended. In other cases a common name is based on the same general principles as the systematic name but with a minimum of detail to produce a name short enough for convenient use. A few names of proteolytic enzymes ending in *-in* are retained; all other enzyme names should end in *-ase*.

(Systematic Names)

Systematic names consist of two parts. The first contains the name of the substrate or in the case of a bimolecular reaction, of the two substrates separated by a colon. The second part ending in *-ase* indicates the nature of the reaction.

Rule 8.

A number of generic words indicating a type of reaction may be used in either common or systematic names: *oxidoreductase, oxygenase, transferase* (with a prefix indicating the nature of the group transferred), *hydrolase, lyase, racemase, epimerase, isomerase, mutase, ligase.*

Rule 9.

(Common Names)

A number of additional generic words indicating reaction types are used in common names but not in the systematic nomenclature *e.g. dehydrogenase, reductase, oxidase, peroxidase, kinase, tautomerase, deaminase, dehydratase etc.*

Rule 10.

Where additional information is needed to make the reaction clear a phrase indicating the reaction or a product should be added in parentheses after the second part of the name *e.g. (ADP-forming), (dimerizing), (CoA-acylating).*

Rule 11.

(Common Names)

The direct attachment of *-ase* to the name of the substrate will indicate that the enzyme brings about hydrolysis.

(Systematic Names)

The suffix *-ase* should never be attached directly to the name of the substrate.

Rule 12.

(Common Names)

The name 'dehydrase', which was at one time used for both dehydrogenating and dehydrating enzymes, should not be used. *Dehydrogenase* will be used for the former and *dehydratase* for the latter.

Rule 13.

(Common Names)

Where possible common names should normally be based on a reaction direction that has been demonstrated *e.g. dehydrogenase* or *reductase*, *decarboxylase* or *carboxylase*.

(Systematic Names)

In the case of reversible reactions the direction chosen for naming should be the same for all the enzymes in a given class even if this direction has not been demonstrated for all. Thus systematic names may be based on a written reaction even though only the reverse of this has been actually demonstrated experimentally.

Rule 14.

(Systematic Names)

When the overall reaction includes two different changes *e.g.* an oxidative demethylation, the classification and systematic name should be based, whenever possible on the one (or the first one) catalyzed by the enzyme; the other function(s) should be indicated by adding a suitable participle in parentheses as in the case of *sarcosine:oxygen oxidoreductase (demethylating)* (EC 1.5.3.1); D-*aspartate:oxygen oxidoreductase (deaminating)* (EC 1.4.3.1); L-*serine hydro-lyase (adding indoleglycerol-phosphate)* (EC 4.2.1.20).

Other examples of such additions are *(decarboxylating), (cyclizing), (acceptor-acylating), (isomerizing).*

Rule 15.

When an enzyme catalyses more than one type of reaction the name should normally refer to one reaction only. Each case must be considered on its merits and the choice must be to some extent arbitrary. Other important activities of the enzyme may be indicated in the List under 'Reaction' or 'Comments'.

Similarly, when any enzyme acts on more than one substrate (or pair of substrates) the name should normally refer only to one substrate (or pair of substrates) although in certain cases it may be possible to use a term that covers a whole group of substrates or an alternative substrate may be given in parentheses.

Rule 16.

A group of enzymes with closely similar specificities should normally be described by a single entry. However when the specificity of two enzymes catalyzing the same reactions is sufficiently different (the degree of difference being a matter of arbitrary choice) two separate entries may be made *e.g.* EC 1.2.1.4 and EC 1.2.1.7. Separate entries are also appropriate for enzymes having similar catalytic functions but known to differ basically with regard to reaction mechanism or to the nature of the catalytic groups *e.g. amine oxidase (flavin-containing)* (EC 1.4.3.4) and *amine oxidase (copper-containing)* (EC 1.4.3.6).

(b) Rules and Guidelines for Particular Classes of Enzymes

Class 1

Rule 17.

(Common Names)

The terms *dehydrogenase* or *reductase* will be used much as hitherto. The latter term is appropriate when hydrogen transfer from the substance mentioned, as donor in the systematic name is not readily demonstrated. *Transhydrogenase* may be retained for a few well-established cases. *Oxidase* is used only for cases where O_2 acts as an acceptor, and *oxygenase* only for those cases where the O_2 molecule (or part of it) is directly incorporated into the substrate. *Peroxidase* is used for enzymes using H_2O_2 as acceptor. *Catalase* must be regarded as exceptional. Where no ambiguity is caused, the second reactant is not usually named; but where required to prevent ambiguity, it may be given in parentheses *e.g.* EC 1.1.1.1, *alcohol dehydrogenase* and EC 1.1.1.2, *alcohol dehydrogenase* ($NADP^+$).

(Systematic Names)

All enzymes catalyzing oxidoreductions should be named *oxidoreductases* in the systematic nomenclature, and the names formed on the pattern *donor:acceptor oxidoreductase.*

Rule 18.

(Systematic Names)

For oxidoreductases using NAD^+ or $NADP^+$ the coenzyme should always be named as the acceptor except for the special case of Section 1.6 (enzymes whose normal physiological function is regarded as reoxidation of the reduced coenzyme). Where the enzyme can use either coenzyme this should be indicated by writing $NAD(P)^+$.

Rule 19.

Where the true acceptor is unknown and the oxidoreductase has only been shown to react with artificial acceptors the word *acceptor* should be written in parentheses as in the case of EC 1.3.99.1, *succinate:(acceptor) oxidoreductase.*

Rule 20.

(Common Names)

Oxidoreductases that bring about the incorporation of molecular oxygen into one donor or into either or both of a pair of donors are named *oxygenase*. If only one atom of oxygen is incorporated the term *monooxygenase* is used; if both atoms of O_2 are incorporated, the term *dioxygenase* is used.

(Systematic Names)

Oxidoreductases bringing about the incorporation of oxygen into one of paired donors should be named on the pattern *donor,donor:oxygen oxidoreductase (hydroxylating).*

Class 2.

Rule 21.

(Common Names)

Only one specific substrate or reaction product is generally indicated in the common names together with the group donated or accepted.

The forms *transaminase etc.* may be replaced if desired by the corresponding forms *aminotransferase etc.*

A number of special words are used to indicate reaction types *e.g. kinase* to indicate a phosphate transfer from ATP to the named substrate (not 'phosphokinase'), *diphosphokinase* for a similar transfer of diphosphate.

(Systematic Names)

Enzymes catalyzing group-transfer reactions should be named *transferase* and the names formed on the pattern *donor:acceptor group-transferred-transferase e.g. ATP:acetate phosphotransferase* (EC 2.7.2.1). A figure may be prefixed to show the position to which the group is transferred, *e.g. ATP:D-fructose 1-phosphotransferase* (EC 2.7.1.3). The spelling 'transphorase' should not be used. In the case of the phosphotransferases, ATP

should always be named as the donor. In the case of the transaminases involving 2-oxoglutarate, the latter should always be named as the acceptor.

Rule 22.

(Systematic Names)

The prefix denoting the group transferred should as far as possible be non-committal with respect to the mechanism of the transfer *e.g. phospho-* rather than *phosphate-*.

Class 3.

Rule 23.

(Common Names)

The direct addition of *-ase* to the name of the substrate generally denotes a hydrolase. Where this is difficult *e.g.* for EC 3.1.2.1, the word *hydrolase* may be used. Enzymes should not normally be given separate names merely on the basis of optimal conditions for activity. The acid and alkaline phosphatases (EC 3.1.3.1-2) should be regarded as special cases and not as examples to be followed. The common name *lysozyme* is also exceptional.

(Systematic Names)

Hydrolysing enzymes should be systematically named on the pattern *substrate hydrolase*. Where the enzyme is specific for the removal of a particular group, the group may be named as a prefix *e.g. adenosine aminohydrolase* (EC 3.5.4.4). In a number of cases this group can also be transferred by the enzyme to other molecules and the hydrolysis itself might be regarded as a transfer of the group to water.

Class 4.

Rule 24.

(Common Names)

The old names *decarboxylase, aldolase etc*. are retained; and *dehydratase* (not 'dehydrase') is used for the hydro-lyases. 'Synthetase' should not be used for any enzymes in this class. The term *synthase* may be used instead for any enzyme in this class (or any other class) when it is desired to emphasize the synthetic aspect of the reaction.

(Systematic Names)

Enzymes removing groups from substrates non-hydrolytically leaving double bonds (or adding groups to double bonds) should be called *lyases* in the systematic nomenclature. Prefixes such as *hydro-, ammonia-* should be used to denote the type of reaction, *e.g. (*S*)-malate hydro-lyase* (EC 4.2.1.2). Decarboxylases should be regarded as *carboxy-lyases*. A hyphen should always be written before *lyase* to avoid confusion with hydrolases, carboxylases, *etc*.

Rule 25.

(Common Names)

Where the equilibrium warrants it or where the enzyme has long been named after a particular substrate, the reverse reaction may be taken as the basis of the name using *hydratase, carboxylase, etc. e.g. fumarate hydratase* for EC 4.2.1.2 (in preference to 'fumarase' which suggests an enzyme hydrolysing fumarate).

(Systematic Names)

The complete molecule, not either of the parts into which it is separated, should be named as the substrate.

The part indicated as a prefix to *-lyase* is the more characteristic and usually, but not always, the smaller of the two reaction products. This may either be the removed (saturated) fragment of the substrate molecule as in *ammonia-, hydro-, thiol-lyases etc*.

or the remaining unsaturated fragment *e.g.* in the case of *carboxy-*, *aldehyde-* or *oxo-acid-lyases*.

Rule 26.

Various subclasses of the lyases include a number of strictly specific or group-specific pyridoxal-5-phosphate enzymes that catalyze *elimination* reactions of β- or γ-substituted α-amino acids. Some closely related pyridoxal-5-phosphate-containing enzymes *e.g. tryptophan synthase* (EC 4.2.1.20) and *cystathionine* β-*synthase* (EC 4.2.1.22) catalyze *replacement* reactions in which a β- or γ- substituent is replaced by a second reactant without creating a double bond. Formally these enzymes appear to be transferases rather than lyases. However there is evidence that in these cases the elimination of the β- or γ-substituent and the formation of an unsaturated intermediate is the first step in the reaction. Thus applying rule 14 these enzymes are correctly classified as lyases.

Class 5.

Rule 27.

In this class, the common names are in general similar to the systematic names which indicate the basis of classification.

Rule 28.

Isomerase will be used as a general name for enzymes in this class. The types of isomerization will be indicated in systematic names by prefixes *e.g. maleate cis-trans-isomerase* (EC 5.2.1.1), *phenylpyruvate keto-enol-isomerase* (EC 5.3.2.1), *3-oxosteroid* Δ^5- Δ^4-*isomerase* (EC 5.3.3.1). Enzymes catalyzing an aldose-ketose interconversion will be known as *ketol-isomerase, e.g.* L-*arabinose ketol-isomerase* (EC 5.3.1.4). When the isomerization consists of an intramolecular transfer of a group the enzyme is named a *mutase e.g.* EC 5.4.1.1 and the *phosphomutases* in sub-subclass 5.4.2; when it consists of

an intramolecular lyase-type reaction *e.g.* EC 5.5.1.1, it is systematically named a *lyase (decyclizing)*.

Rule 29.

Isomerases catalyzing inversions at asymmetric centers should be termed *racemases* or *epimerases* according to whether the substrate contains one or more than one center of asymmetry. Compare, for example EC 5.1.1.5 with EC 5.1.1.7. A numerical prefix to the word *epimerase* should be used to show the position of the inversion.

Class 6.

Rule 30

(Common Names)

Common names for enzymes of this class were previously of the type *XY synthetase*. However as this use has not always been understood and synthetase has been confused with synthase it is now recommended that as far as possible the common names should be similar in form to the systematic names.

(Systematic Names)

The class of enzymes catalyzing the linking together of two molecules coupled with the breaking of a diphosphate link in ATP *etc*. should be known as *ligases*. These enzymes were often previously known as 'synthetases'; however this terminology differs from all other systematic enzyme names in that it is based on the product and not on the substrate. For these reasons a new systematic class name was necessary.

Rule 31

(Common Names)

The common names should be formed on the pattern *X-Y ligase*, where X-Y is the substance formed by linking X and Y. In certain cases, where a trivial name is commonly

used for XY a name of the type *XY synthase* may be recommended (*e.g.* EC 6.3.2.11, *carnosine synthase*).

(Systematic Names)

The systematic names should be formed on the pattern *X:Y ligase (ADP-forming)*, where X and Y are the two molecules to be joined together. The phrase shown in parentheses indicates both that ATP is involved and also that the terminal diphosphate link is broken. Thus the reaction is X + Y + ATP = X-Y + ADP + P_i.

Rule 32.

(Common Names)

In special, when glutamine acts as an ammonia-donor it is indicated by adding in parentheses (*glutamine-hydrolysing*) to a ligase name.

(Systematic Names)

In this case the name *amido-ligase* should be used in the systematic nomenclature.

Enzyme Subclasses

A short description of enzyme subclasses is given in Table 6.

Table 6

Enzyme sub-classes

Subclass	Name
EC 1	**Oxidoreductases**
EC 1.1	Acting on the CH-OH group of donors
EC 1.2	Acting on the aldehyde or oxo group of donors
EC 1.3	Acting on the CH-CH group of donors
EC 1.4	Acting on the CH-NH_2 group of donors

EC 1.5	Acting on the CH-NH group of donors
EC 1.6	Acting on NADH or NADPH
EC 1.7	Acting on other nitrogenous compounds as donors
EC 1.8	Acting on a sulfur group of donors
EC 1.9	Acting on a heme group of donors
EC 1.10	Acting on diphenols and related substances as donors
EC 1.11	Acting on a peroxide as acceptor
EC 1.12	Acting on hydrogen as donor
EC 1.13	Acting on single donors with incorporation of molecular oxygen (oxygenases)
EC 1.14	Acting on paired donors, with incorporation or reduction of molecular oxygen
EC 1.15	Acting on superoxide radicals as acceptor
EC 1.16	Oxidising metal ions
EC 1.17	Acting on CH or CH_2 groups
EC 1.18	Acting on iron-sulfur proteins as donors
EC 1.19	Acting on reduced flavodoxin as donor
EC 1.20	Acting on phosphorus or arsenic in donors
EC 1.21	Acting on X-H and Y-H to form an X-Y bond
EC 1.97	Other oxidoreductases
EC 2	**Transferases**
EC 2.1	Transferring one-carbon groups
EC 2.2	Transferring aldehyde or ketonic groups
EC 2.3	Acyltransferases
EC 2.4	Glycosyltransferases
EC 2.5	Transferring alkyl or aryl groups, other than methyl groups
EC 2.6	Transferring nitrogenous groups
EC 2.7	Transferring phosphorus-containing groups
EC 2.8	Transferring sulfur-containing groups

EC 2.9	Transferring selenium-containing groups
EC 3	**Hydrolases**
EC 3.1	Acting on ester bonds
EC 3.2	Glycosylases
EC 3.3	Acting on ether bonds
EC 3.4	Acting on peptide bonds (peptidases)
EC 3.5	Acting on carbon-nitrogen bonds, other than peptide bonds
EC 3.6	Acting on acid anhydrides
EC 3.7	Acting on carbon-carbon bonds
EC 3.8	Acting on halide bonds
EC 3.9	Acting on phosphorus-nitrogen bonds
EC 3.10	Acting on sulfur-nitrogen bonds
EC 3.11	Acting on carbon-phosphorus bonds
EC 3.12	Acting on sulfur-sulfur bonds
EC 3.13	Acting on carbon-sulfur bonds
EC 4	**Lyases**
EC 4.1	Carbon-carbon lyases
EC 4.2	Carbon-oxygen lyases
EC 4.3	Carbon-nitrogen lyases
EC 4.4	Carbon-sulfur lyases
EC 4.5	Carbon-halide lyases
EC 4.6	Phosphorus-oxygen lyases
EC 4.99	Other lyases
EC 5	**Isomerases**
EC 5.1	Racemases and epimerases
EC 5.2	*cis-trans*-Isomerases
EC 5.3	Intramolecular isomerases

EC 5.4	Intramolecular transferases (mutases)
EC 5.5	Intramolecular lyases
EC 5.99	Other isomerases
EC 6	**Ligases**
EC 6.1	Forming carbon—oxygen bonds
EC 6.2	Forming carbon—sulfur bonds
EC 6.3	Forming carbon—nitrogen bonds
EC 6.4	Forming carbon—carbon bonds
EC 6.5	Forming phosphoric ester bonds
EC 6.6	Forming nitrogen—metal bonds

Detailed level wise information can be obtained by visiting the URL http://www.genome.jp/htbin/get_htext?ECtable.

COMPARISON OF NORMAL AND KEGG METABOLIC MAPS

Traditional textbook representations of metabolic pathways are totally different from that of metabolic pathway databases. While the textbook version is solidly supported by proofs from laborious, time consuming wet-lab experiments, data of electronic databases are rather based on logic and prediction. ~90% of these data are mathematically derived.

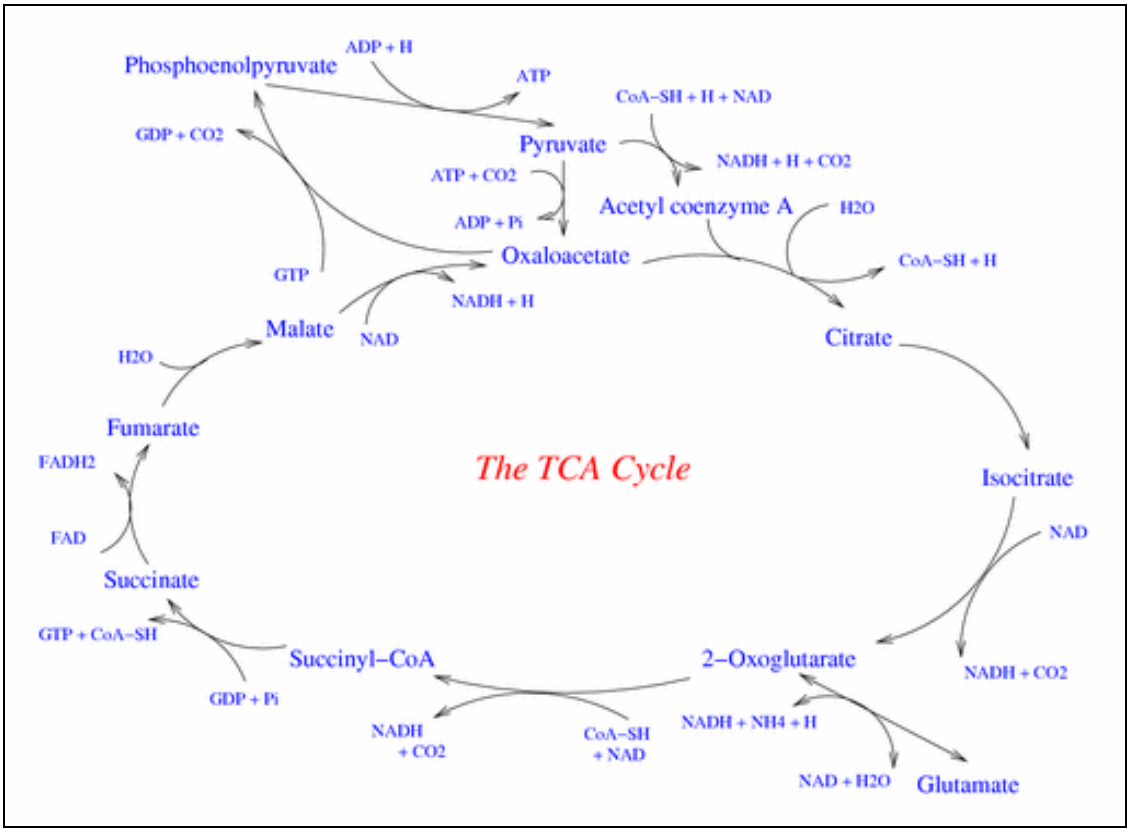
For example the Citric Acid Cycle explained in books is a simple affair of eight steps involving aldol condensation with a number of oxidations in presence of cofactors like Thiamine Pyrophosphate(TPP), Lipoamide, FAD, NAD^+, CoA, producing energy as well as reduction power($NADH + H^+$) .

But they neither capture the full number of potential network functions nor the network's susceptibleness to disruption. For these reasons only many aspects of metabolic network functions remain unclear. For example, in stoichiometric network analysis, it is

convenient to define a subset of central metabolic intermediates that represent the products of catabolism that are used to initiate anabolism. However, even for *Escherichia coli*, there is no agreement on the identity of this subset.

Diagram 6

Outline of TCA Cycle



But in KEGG the computerized diagram is not intended to capture the consensus pathway or an organism specific pathway. It represents a collection of all chemically feasible reaction pathways. Thus the actual pathway in a cell or of a certain organism should correspond to a subset or a part of the diagram.

Computation of pathways in KEGG is based on two types of binary relations, *i.e.*, substrate-product relation (extracted from LIGAND database) and enzyme-enzyme relation, which corresponds to a pair of enzymes connected in the actual pathways (the nearest neighbour enzymes). But the maps, not necessarily corresponds to the successive flow of catalytic reactions that actually happens in living organisms. It only captures the overall architecture of metabolic pathways.

Each cycle in the database is represented as a map with:

- Substrates and Products as nodes represented by circles along with their names (in a metabolic pathway product of one reaction acts as substrate of the successive reaction),

- Relation between nodes as solid arrows accompanied with enzyme commission numbers in rectangular boxes, representing each reaction according to its nature.

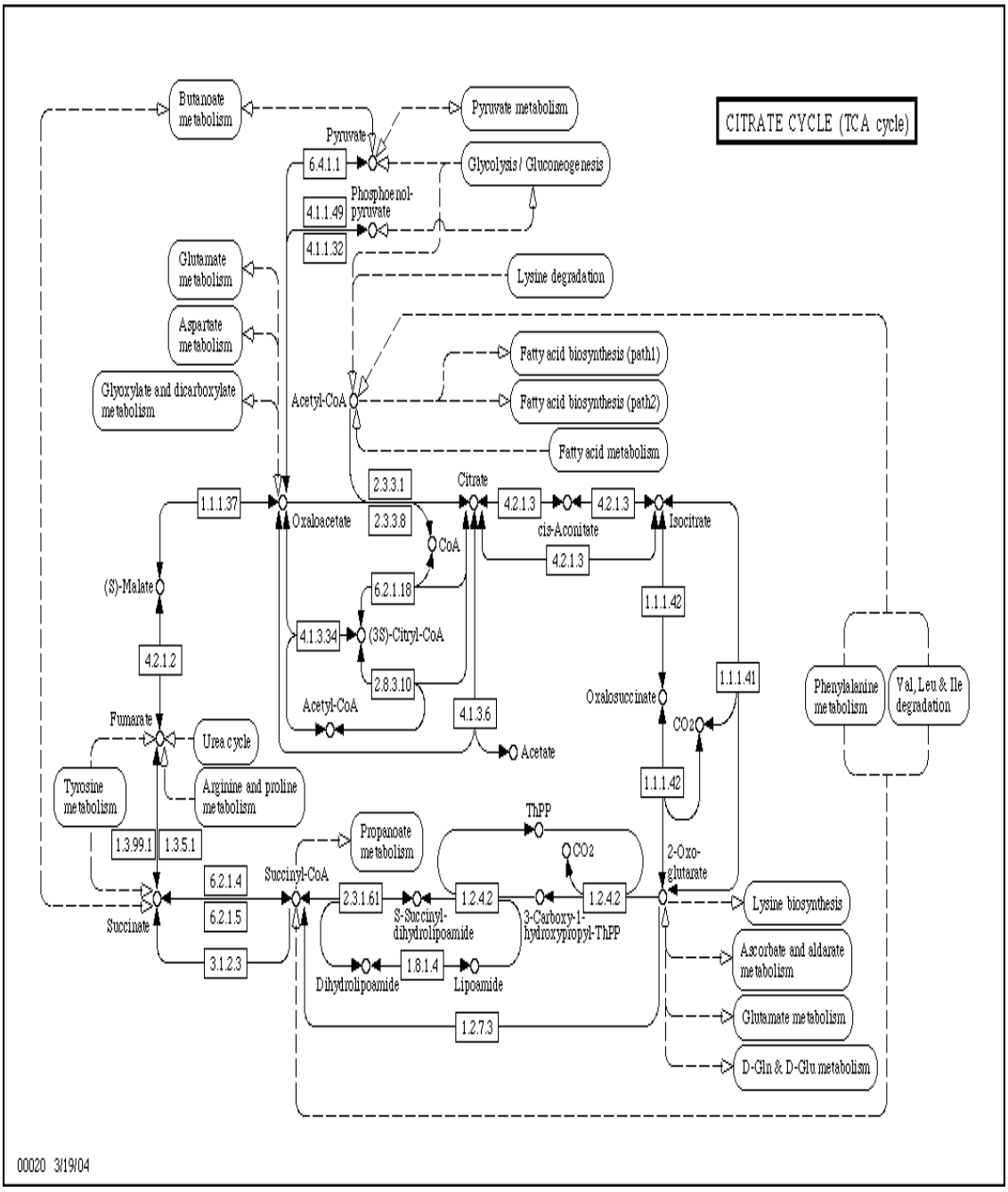
- The subset of the map featuring solid arrows is the superset of citric acid cycles present in different organisms, representing the generalized view of citric acid cycle.

In contrary the dotted portions represent link of the nodes present in citric acid cycle with other metabolic pathways involving carbohydrates, amino acids, fats etc. For example in diagram 2:

1. Pyruvate is related to glycolysis, gluconeogenesis, pyruvate and butanoate metabolism
2. Phosphoenol pyruvate is linked to glycolysis and gluconeogenesis

Diagram 7

KEGG- TCA Cycle

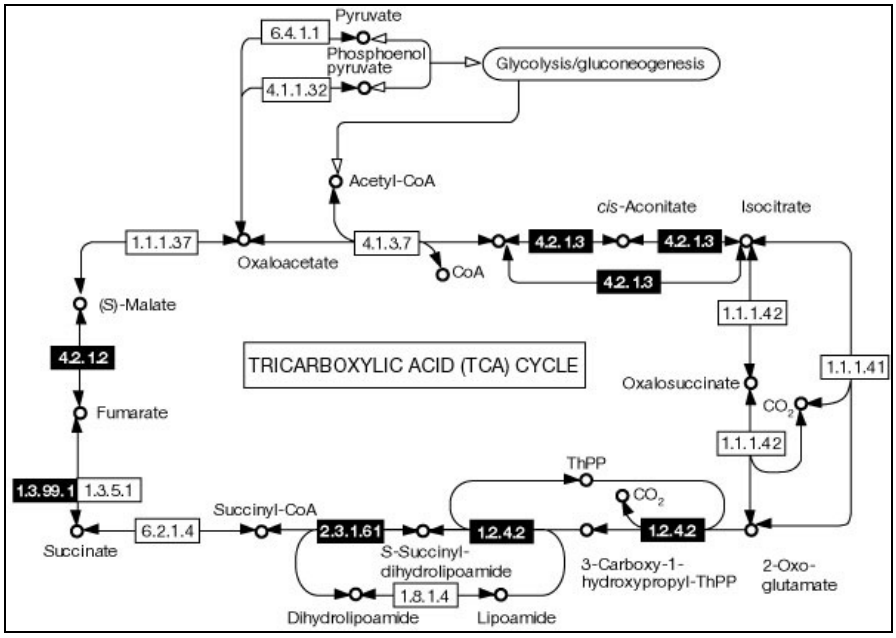


3. Acetyl CoA being the versatile cofactor is part of glycolysis, gluconeogenesis, lysine degradation, phenylalanine metabolism, Val, Leu and Ile degradation, fatty acid biosynthesis (path1 and Path2), fatty acid metabolism etc.
4. Oxaloacetate is part of glutamate, aspartate, glyoxylate and dicarboxylate metabolism
5. 2-Oxo glutarate is linked to lysine biosynthesis, ascorbate, aldarate, glutamate, D-Gln and D-Glu metabolism
6. Succinyl CoA is related to propanoate and phenylalanine metabolism, Val, Leu, Ile degradation,
7. Succinate takes role in tyrosine and butanoate metabolism
8. Fumarate is part of Urea cycle, arginine and proline metabolism

- Species specific maps are designed by marking the enzyme commission numbers corresponding to active reactions with different colors.

Diagram 8

TCA Cycle of mouse genome



This is the mouse metabolic diagram of the tricarboxylic acid (TCA) cycle. Enzymes are represented in boxes with their corresponding Enzyme Commission (EC) numbers and are connected with their metabolites. EC numbers in black boxes are found only in the FANTOM2 clone set while the unshaded boxes are part of general dataset for mouse tricarboxylic acid cycle.

EC Numbers used in KEGG-TCA Cycle

(a)- **6. Ligases**

6.4 Forming carbon-carbon bonds

6.4.1 Forming carbon-carbon bonds

6.4.1.1 pyruvate carboxylase; pyruvic carboxylase

(b)- **4. Lyases**

4.1 Carbon-carbon lyases

4.1.1 Carboxy-lyases

4.1.1.49 phosphoenolpyruvate carboxykinase (ATP); phosphopyruvate carboxylase (ATP); phosphoenolpyruvate carboxylase; phosphoenolpyruvate carboxykinase; phosphopyruvate carboxykinase (adenosine triphosphate); PEP carboxylase; PEP carboxykinase; PEPCK (ATP); PEPK; PEPCK; phosphoenolpyruvic carboxylase; phosphoenolpyruvic carboxykinase; phosphoenolpyruvate carboxylase (ATP); phosphopyruvate carboxykinase

©- **4. Lyases**

4.1 Carbon-carbon lyases

4.1.1 Carboxy-lyases

4.1.1.32 phosphoenolpyruvate carboxykinase (GTP); phosphoenolpyruvate carboxylase; phosphopyruvate carboxylase; phosphopyruvate (guanosine triphosphate) carboxykinase; phosphoenolpyruvic carboxykinase (GTP); phosphopyruvate carboxylase

(GTP); phosphoenolpyruvic carboxylase (GTP); phosphoenolpyruvic carboxykinase; phosphoenolpyruvate carboxykinase; PEP carboxylase

(h)- **6. Ligases**

6.2 Forming carbon-sulfur bonds

6.2.1 Acid--thiol ligases

6.2.1.18 citrate-CoA ligase

(i)- **4. Lyases**

4.1 Carbon-carbon lyases

4.1.3 Oxo-acid-lyases

4.1.3.6 citrate (pro-3S)-lyase; citrase; citratase; citritase; citridesmolase; citrate aldolase; citric aldolase; citrate lyase; citrate oxaloacetate-lyase

(j)- **4. Lyases**

4.2 Carbon-oxygen lyases

4.2.1 Hydro-lyases

4.2.1.3 aconitate hydratase; cis-aconitase; aconitase

(k)- **1. Oxidoreductases**

1.1 Acting on the CH-OH group of donors

1.1.1 With NAD+ or NADP+ as acceptor

1.1.1.41 isocitrate dehydrogenase (NAD+); isocitric dehydrogenase; beta-ketoglutaric-isocitric carboxylase; isocitric acid dehydrogenase; NAD dependent isocitrate dehydrogenase; NAD isocitrate dehydrogenase; NAD-linked isocitrate dehydrogenase; NAD-specific isocitrate dehydrogenase; NAD isocitric dehydrogenase; isocitrate dehydrogenase (NAD)

(l)- **1. Oxidoreductases**

1.1 Acting on the CH-OH group of donors

1.1.1 With NAD+ or NADP+ as acceptor

1.1.1.42 isocitrate dehydrogenase (NADP+); oxalosuccinate decarboxylase; isocitrate dehydrogenase (NADP); oxalsuccinic decarboxylase; isocitrate (NADP) dehydrogenase; isocitrate (nicotinamide adenine dinucleotide phosphate) dehydrogenase; NADP-specific isocitrate dehydrogenase; NADP-linked isocitrate dehydrogenase; NADP-dependent isocitrate dehydrogenase; NADP isocitric dehydrogenase; isocitrate dehydrogenase (NADP-dependent); NADP- dependent isocitric dehydrogenase; NADP+-linked isocitrate dehydrogenase

(m)- **1. Oxidoreductases**

1.2 Acting on the aldehyde or oxo group of donors

1.2.4 With a disulfide as acceptor

1.2.4.2 oxoglutarate dehydrogenase (succinyl-transferring); 2- ketoglutarate dehydrogenase; 2-oxoglutarate dehydrogenase; 2- oxoglutarate: lipoate oxidoreductase; 2-oxoglutarate:lipoamide 2- oxidoreductase (decarboxylating and acceptor-succinylating); alpha-ketoglutarate dehydrogenase; alphaketoglutaric acid dehydrogenase; alpha-ketoglutaric dehydrogenase; alpha- oxoglutarate dehydrogenase; AKGDH; OGDC; ketoglutaric dehydrogenase; oxoglutarate decarboxylase; oxoglutarate dehydrogenase; oxoglutarate dehydrogenase (lipoamide)

(n)- 2. Transferases

2.3 Acyltransferases

2.3.1 Transferring groups other than amino-acyl groups

2.3.1.61 dihydrolipoyllysine-residue succinyltransferase; dihydrolipoamide S-succinyltransferase; dihydrolipoamide succinyltransferase; dihydrolipoic transsuccinylase; dihydrolipolyl transsuccinylase; dihydrolipoyl transsuccinylase; lipoate succinyltransferase (Escherichia coli); lipoic transsuccinylase; lipoyl transsuccinylase;

succinyl- CoA:dihydrolipoamide S-succinyltransferase; succinyl- CoA:dihydrolipoate S-succinyltransferase

(o)- **1. Oxidoreductases**

1.8 Acting on a sulfur group of donors

1.8.1 With NAD+ or NADP+ as acceptor

1.8.1.4 dihydrolipoyl dehydrogenase; LDP-Glc; LDP-Val; dehydrolipoate dehydrogenase; diaphorase; dihydrolipoamide dehydrogenase; dihydrolipoamide:NAD+ oxidoreductase; dihydrolipoic dehydrogenase; dihydrothioctic dehydrogenase; lipoamide dehydrogenase (NADH); lipoamide oxidoreductase (NADH); lipoamide reductase; lipoamide reductase (NADH2); lipoate dehydrogenase; lipoic acid dehydrogenase; lipoyl dehydrogenase

(p)- **1. Oxidoreductases**

1.2 Acting on the aldehyde or oxo group of donors

1.2.7 With an iron-sulfur protein as acceptor

1.2.7.3 2-oxoglutarate synthase; alpha-ketoglutarate synthase; alpha- ketoglutarate-ferredoxin oxidoreductase; 2-oxoglutarate- ferredoxin oxidoreductase; oxoglutarate synthase

(q)- **6. Ligases**

6.2 Forming carbon-sulfur bonds

6.2.1 Acid--thiol ligases

6.2.1.4 succinate-CoA ligase (GDP-forming); succinyl-CoA synthetase (GDP-forming); succinyl coenzyme A synthetase (guanosine diphosphate- forming); succinate thiokinase; succinic thiokinase; succinyl coenzyme A synthetase; succinate-phosphorylating enzyme; P- enzyme; SCS; G-STK; succinyl coenzyme A synthetase (GDP-forming); succinyl CoA synthetase

(r)- **6. Ligases**

6.2 Forming carbon-sulfur bonds

6.2.1 Acid--thiol ligases

6.2.1.5 succinate-CoA ligase (ADP-forming); succinyl-CoA synthetase (ADP-forming); succinic thiokinase; succinate thiokinase; succinyl-CoA synthetase; succinyl coenzyme A synthetase (adenosine diphosphate-forming); succinyl coenzyme A synthetase; A-STK (adenin nucleotide-linked succinate thiokinase); STK; A-SCS

(s)- **3. Hydrolases**

3.1 Acting on ester bonds

3.1.2 Thiolester hydrolases

3.1.2.3 succinyl-CoA hydrolase; succinyl-CoA acylase; succinyl coenzyme A hydrolase; succinyl coenzyme A deacylase

(t)- **1. Oxidoreductases**

1.3 Acting on the CH-CH group of donors

1.3.99 With other acceptors

1.3.99.1 succinate dehydrogenase; succinic dehydrogenase; fumarate reductase; fumaric hydrogenase; succinodehydrogenase; succinic acid dehydrogenase; succinate oxidoreductase; succinyl dehydrogenase

(u)- **1. Oxidoreductases**

1.3 Acting on the CH-CH group of donors

1.3.5 With a quinone or related compound as acceptor

1.3.5.1 succinate dehydrogenase (ubiquinone); succinic dehydrogenase; complex II; menaquinol: fumarate oxidoreductase; fumarate reductase complex (i.e. FRD, involved in anaerobic respiration, repressed in aerobic respiration); succinate dehydrogenase complex (i. e. SDH, involved in aerobic respiration, repressed in anaerobic respiration)

(v)- **4. Lyases**

4.2 Carbon-oxygen lyases

4.2.1 Hydro-lyases

4.2.1.2 fumarate hydratase; fumarase; L-malate hydro-lyase

(w)- **1. Oxidoreductases**

1.1 Acting on the CH-OH group of donors

1.1.1 With NAD+ or NADP+ as acceptor

1.1.1.37 malate dehydrogenase; malic dehydrogenase; L-malate dehydrogenase; NAD-L-malate dehydrogenase; malic acid dehydrogenase; NAD-dependent malic dehydrogenase; NAD-malate dehydrogenase; NAD-malic dehydrogenase; malate (NAD) dehydrogenase; NAD-dependent malate dehydrogenase; NAD-specific malate dehydrogenase; NAD-linked malate dehydrogenase; MDH; L- malate-NAD+ oxidoreductase

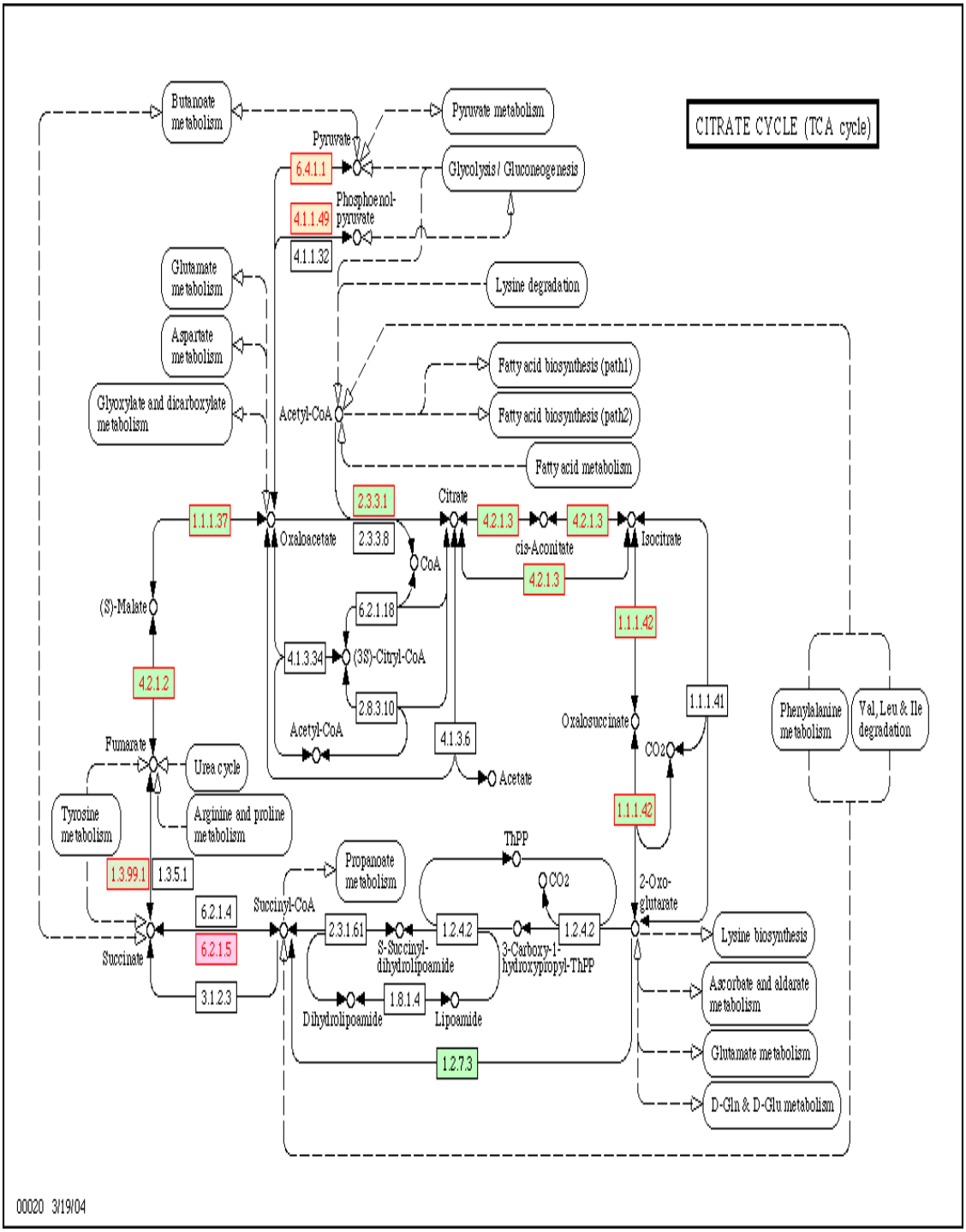
WORK RELATED TO HEYMAN'S AND SINGH'S ALGORITHM

I. SPECIES SELECTED:

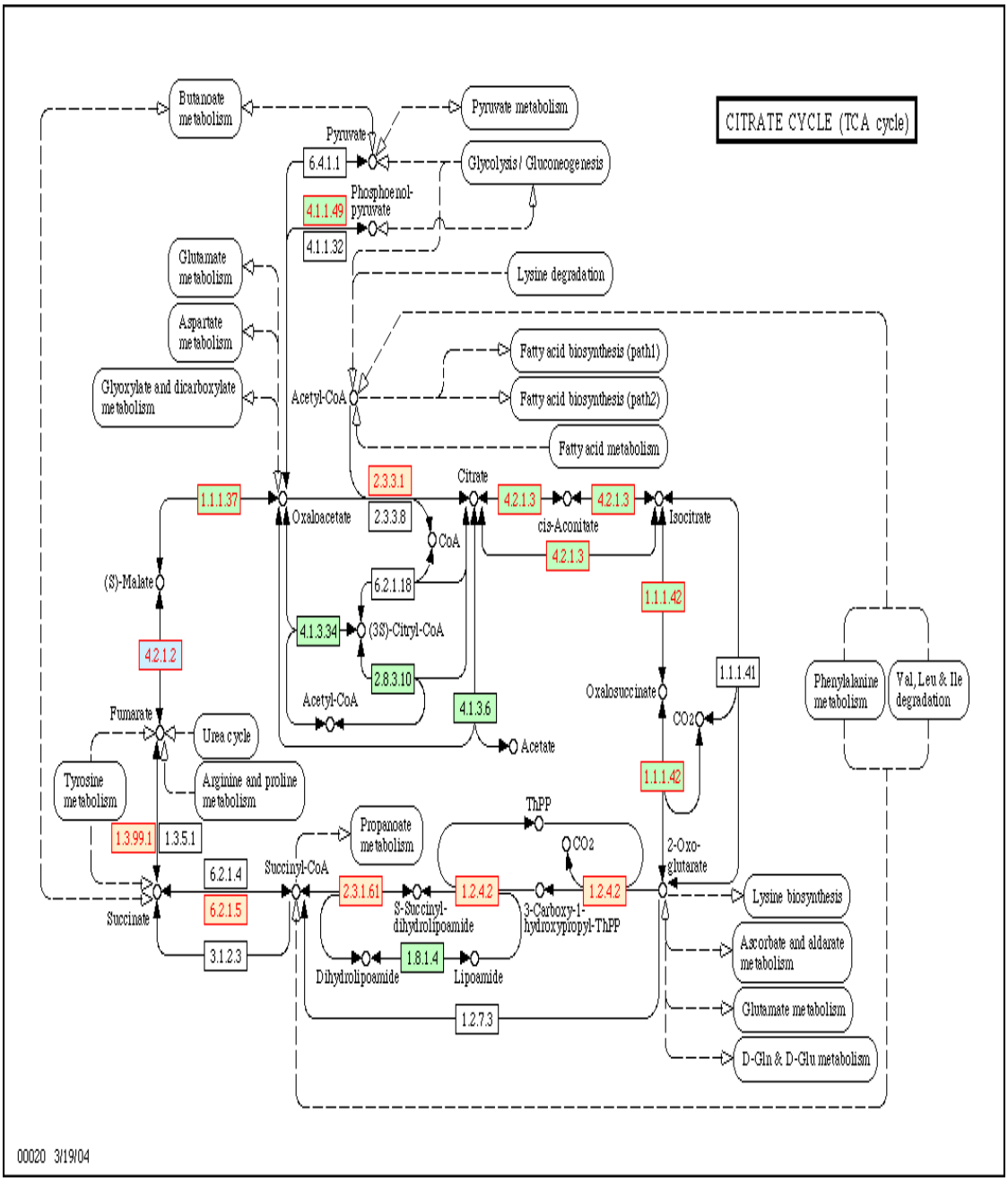
- Campylobacter jejuni,
- Escherchia coli,
- Haemophilus influenzae,
- Photorhabdus luminescence,
- Wigglesworthia brevipalpis ,
- Xanthomonas campestris,
- Yersinia pestis.

II. INPUT MAPS:

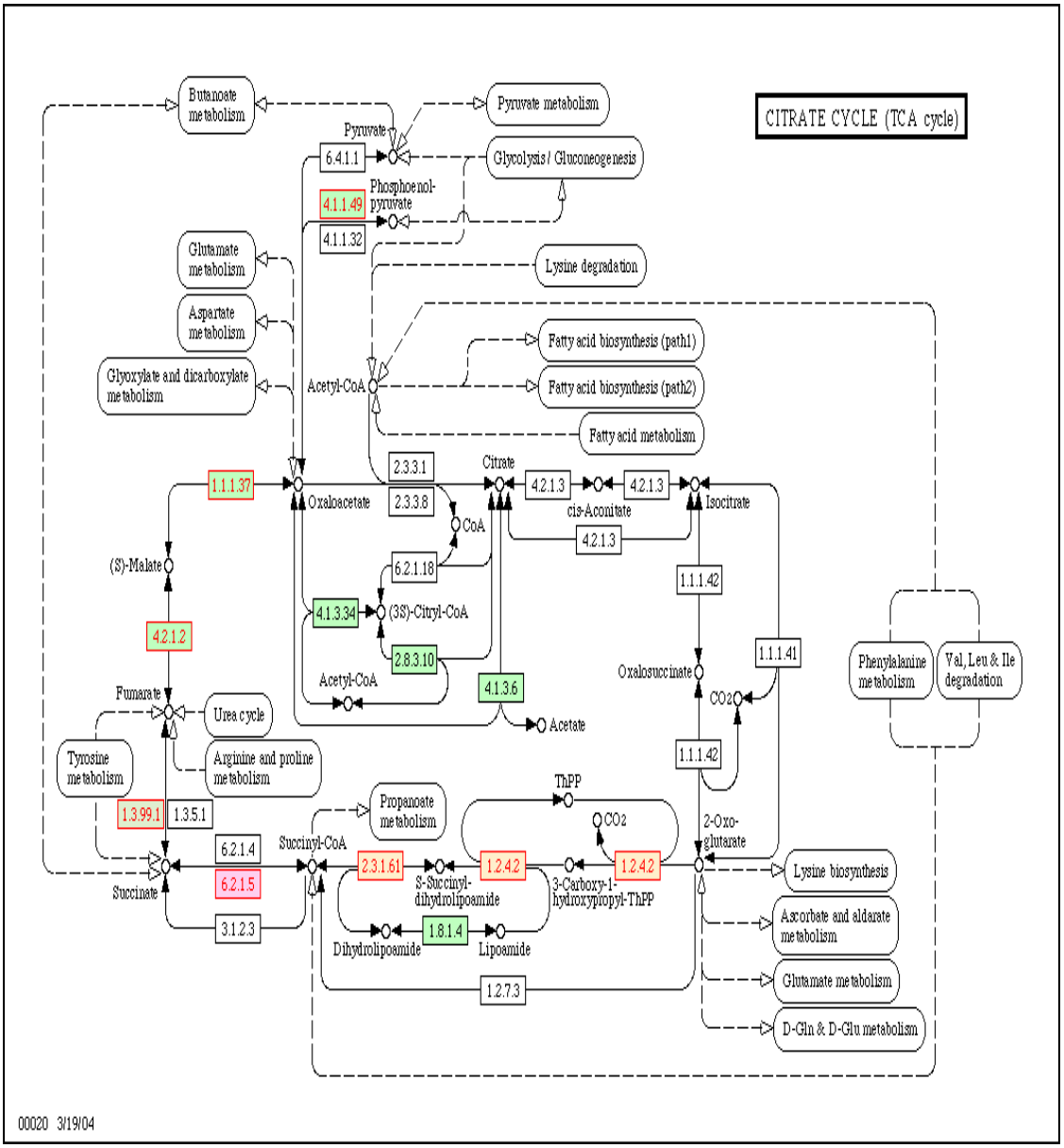
Citrate cycle (TCA cycle) - Campylobacter jejuni


CITRATE CYCLE (TCA cycle)
Butanoate metabolism
Pyruvate metabolism
Pyruvate
6.4.1.1
Glycolysis / Gluconeogenesis
Phosphoenol-pyruvate
4.1.1.49
4.1.1.32
Glutamate metabolism
Lysine degradation
Aspartate metabolism
Glyoxylate and dicarboxylate metabolism
Acetyl-CoA
Fatty acid biosynthesis (path1)
Fatty acid biosynthesis (path2)
Fatty acid metabolism
2.3.3.1
Citrate
1.1.1.37
Oxaloacetate
2.3.3.8
4.2.1.3
4.2.1.3
cis-Aconitate
Isocitrate
CoA
4.2.1.3
(S)-Malate
6.2.1.18
1.1.1.42
4.1.3.34
(3S)-Citryl-CoA
4.2.1.2
2.8.3.10
1.1.1.41
Oxalosuccinate
Phenylalanine metabolism
Val, Leu & Ile degradation
Acetyl-CoA
4.1.3.6
Fumarate
Urea cycle
CO2
Acetate
Tyrosine metabolism
Arginine and proline metabolism
1.1.1.42
ThPP
1.3.99.1
1.3.5.1
Propanoate metabolism
CO2
6.2.1.4
Succinyl-CoA
2-Oxo-glutarate
2.3.1.61
1.2.4.2
1.2.4.2
Lysine biosynthesis
6.2.1.5
S-Succinyl-dihydrolipoamide
3-Carboxy-1-hydroxypropyl-ThPP
Succinate
Ascorbate and aldarate metabolism
3.1.2.3
1.8.1.4
Dihydrolipoamide
Lipoamide
Glutamate metabolism
1.2.7.3
D-Gln & D-Glu metabolism
00020 3/19/04

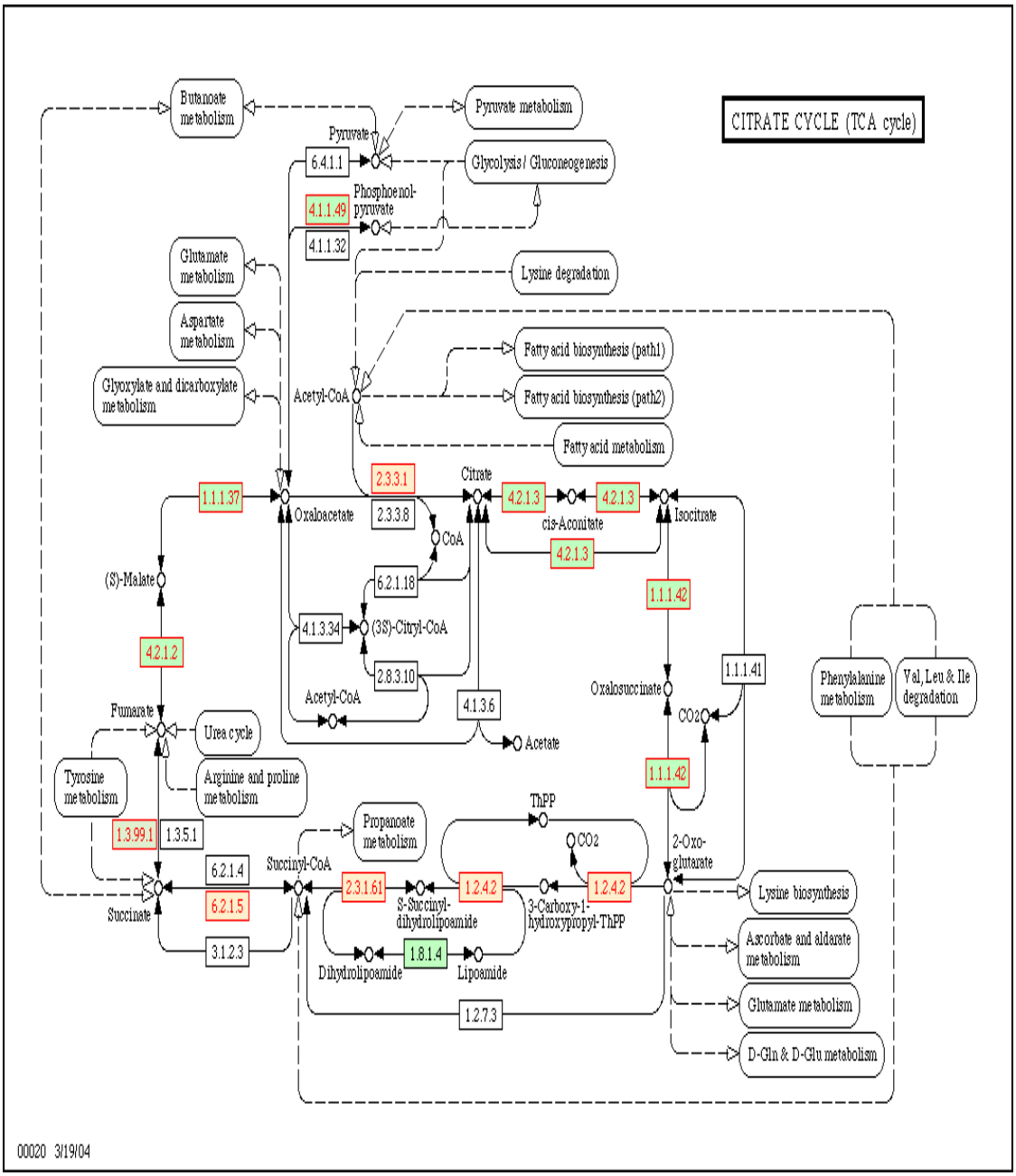
Citrate cycle (TCA cycle) - Escherichia coli K-12 MG1655



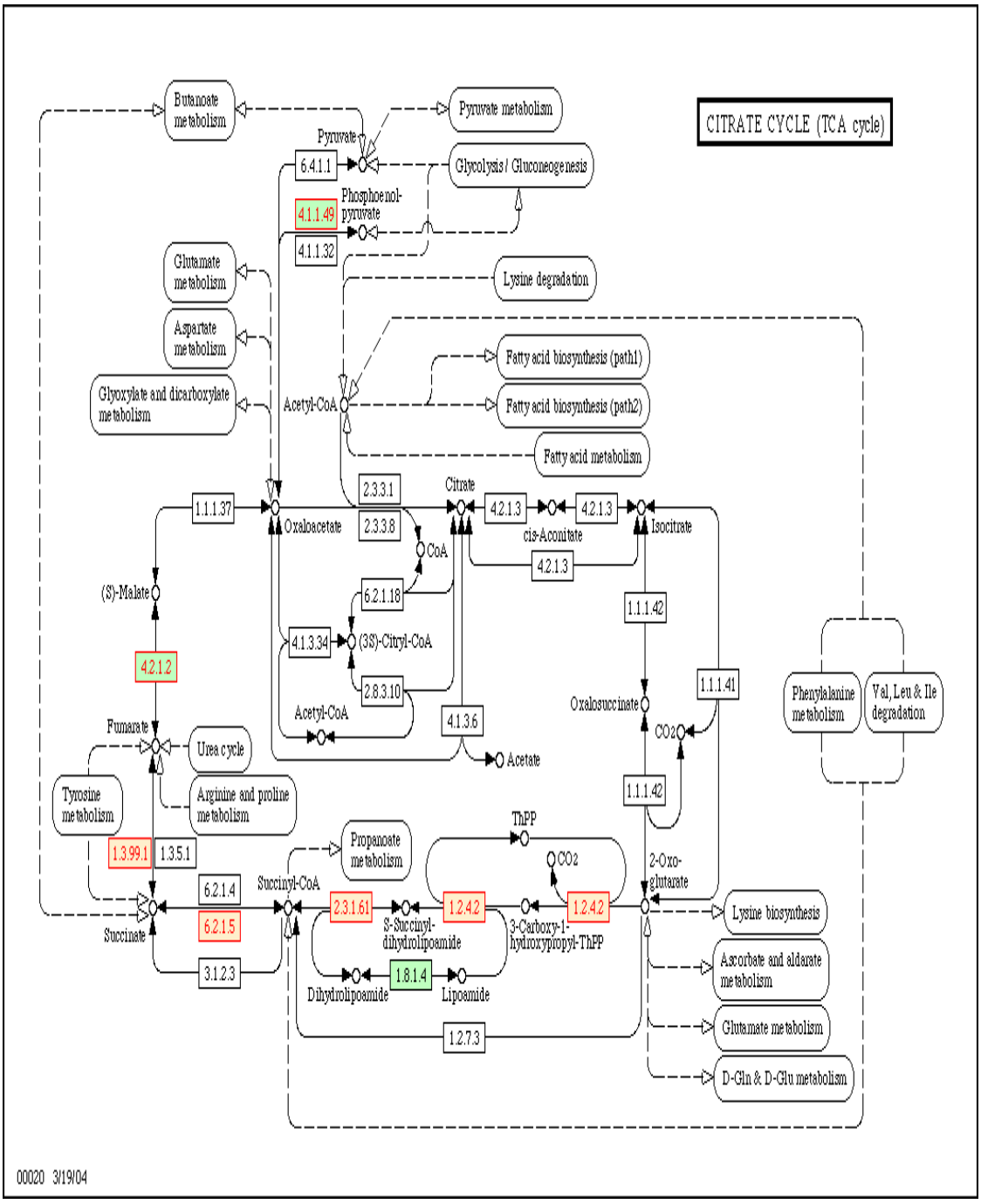
Citrate cycle (TCA cycle) - Haemophilus influenzae



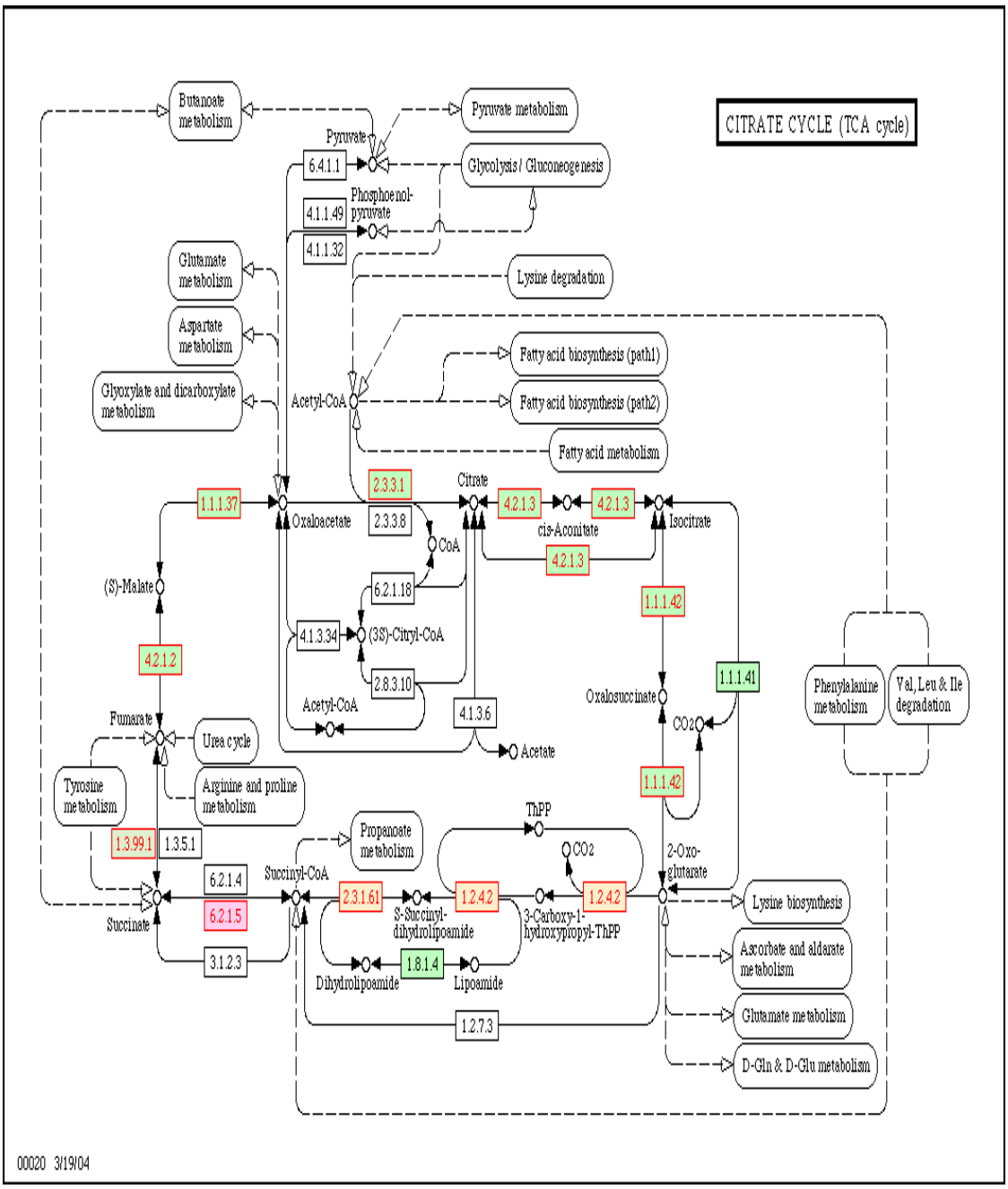
Citrate cycle (TCA cycle) - Photorhabdus luminescens



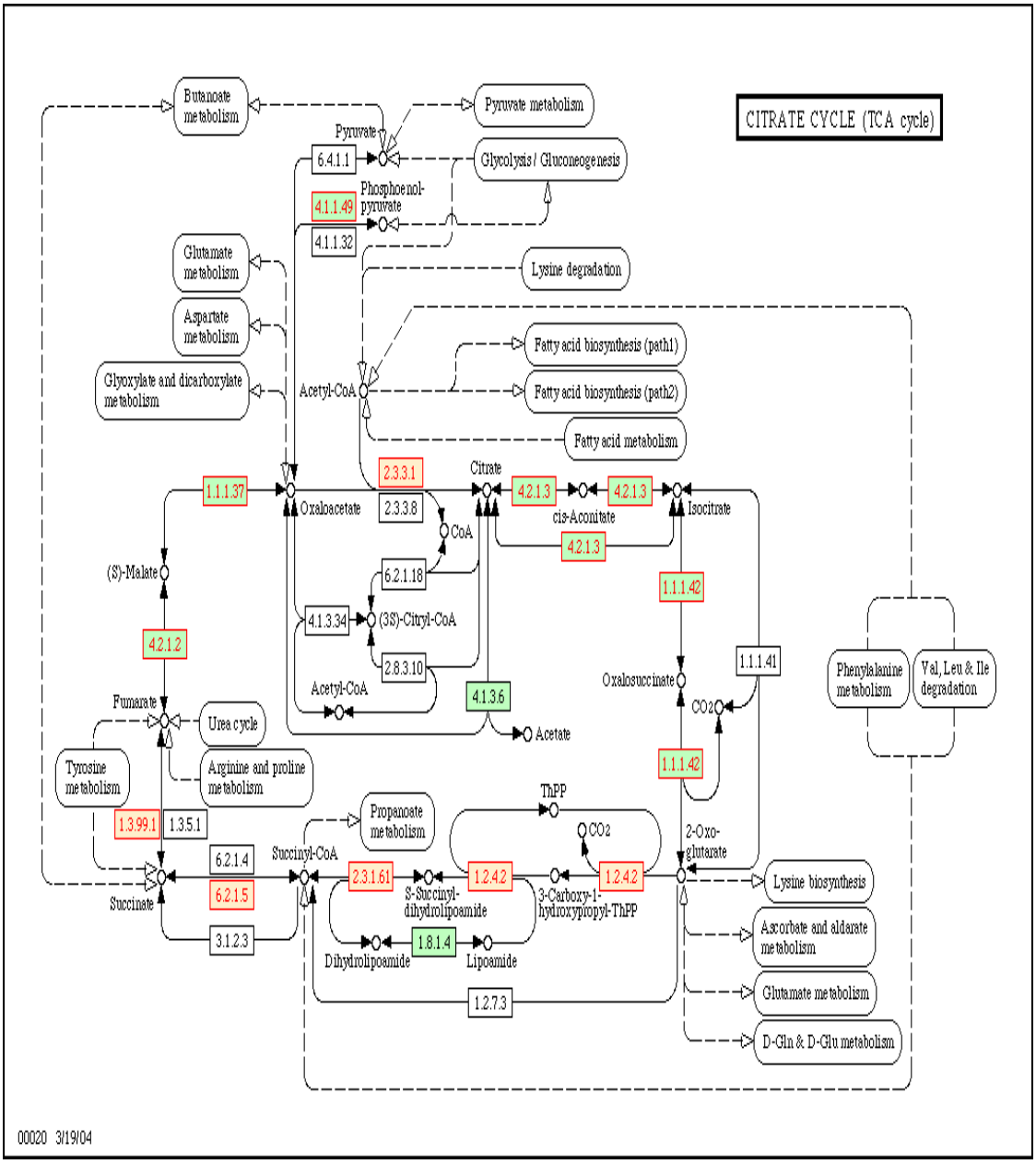
Citrate cycle (TCA cycle) - Wigglesworthia brevipalpis



Citrate cycle (TCA cycle) - Xanthomonas campestris

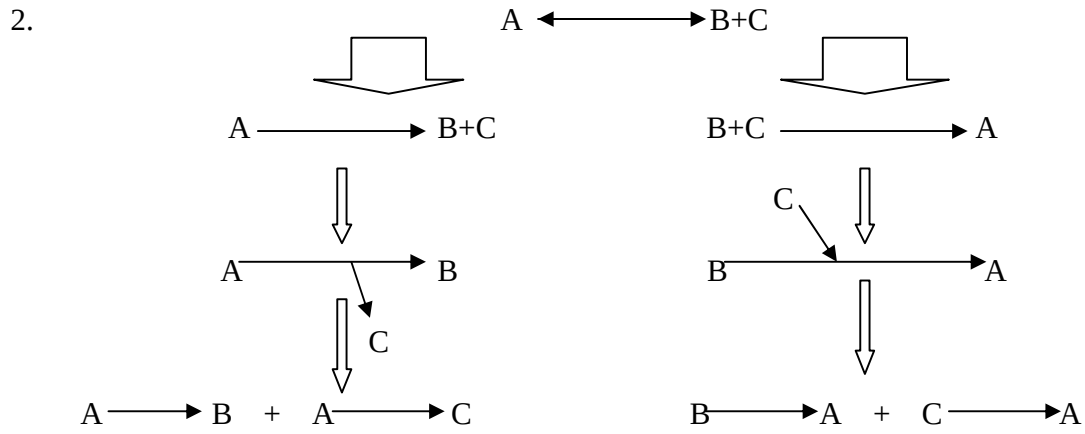


Citrate cycle (TCA cycle) - Yersinia pestis



III. RULES FOR CONVERSION OF MAPS INTO GRAPHS:

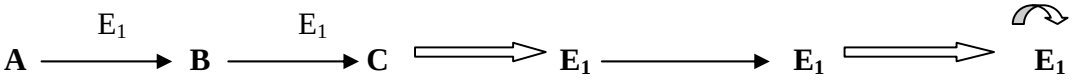
1. Avoid release of CO_2 and H_2O,

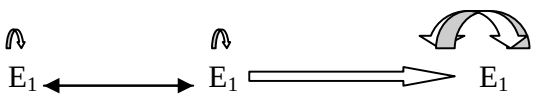
2. 

So $A \longleftrightarrow B + C$

$A \longrightarrow B +$
$B \longrightarrow A +$
$A \longrightarrow C +$
$C \longrightarrow A$

In no means B and C are connected.

3. 

4. 

IV. ASSUMPTIONS FOR SUBSTRATES:

1. A… Pyruvate,
2. B… Phohphoenol pyruvate,
3. C… Acetyl Co A,
4. D… Co A,
5. E… Oxaloacetate,
6. F… Citrate,
7. G… (3S)-Citryl Co A,
8. I… Acetate,
9. J… Cisaconitate,
10. K… Isocitrate,
11. L… Oxaloacetate,
12. M… 2-Oxo-glutarate,
13. N… 3-carboxy-1-hydroxypropyl-Thpp
14. O… S-succinyl Co A
15. P… ThPP
16. Q… Succinyl Co A,
17. R… Dihydrolipoamide,
18. S… Lipoamide,
19. T… Succinate,
20. U… Fumerate,
21. V… (S)-Malate.

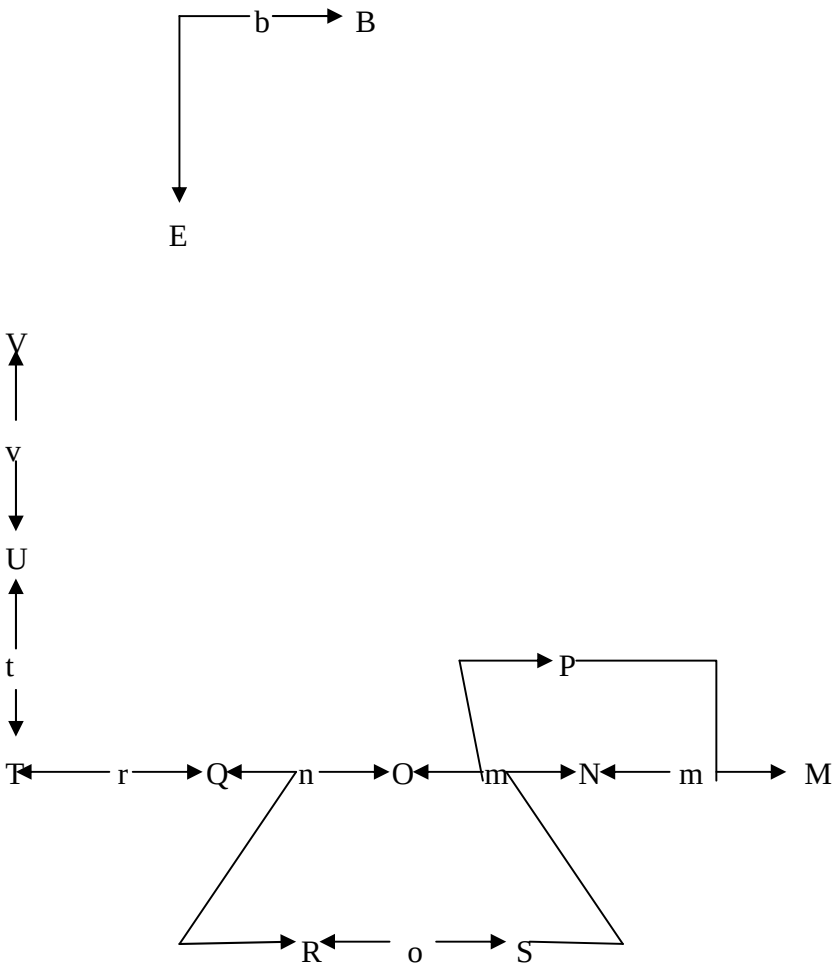
V. ASSUMPTIONS FOR EC NUMBERS:

a… 6.4.1.1,
b… 4.1.1.49,
c… 4.1.1.32,
d… 2.3.3.1,
e… 2.3.3.8,
f… 4.1.3.34,
g… 2.8.3.10,
h… 6.2.1.18,
i… 4.1.3.6,
j… 4.2.1.3,
k… 1.1.1.41,
l… 1.1.1.42,
m… 1.2.4.2,
n… 2.3.1.61,
o… 1.8.1.4,
p… 1.2.7.3,
q… 6.2.1.4,
r… 6.2.1.5,
s… 3.1.2.3,
t… 1.3.99.1,
u… 1.3.5.1,
v… 4.2.1.2,
w… 1.1.1.37.

VI. HOW TO CONSTRUCT THE ENZYME GRAPHS?

1. From a species specific metabolic map, only the coloured part, *i.e*, active reactions are taken separately. For example from the TCA Cycle map of *Wigglesworthia brevipalpis* the following active reactions are taken.

Each created enzyme graph can be represented as G = (V, E) where V is the set of vertices (nodes) and $E \subseteq V \times V$ is the set of edges. If $V = \phi$, then G is empty graph. If for every edge (a, b) $\in$ E there exists the edge (b, a) $\in$ E, then the graph is said to be undirected.



2. In second step, reactions are separated according to their reversibility or irreversibility.

b

B ⟶ E

b

E ⟶ B

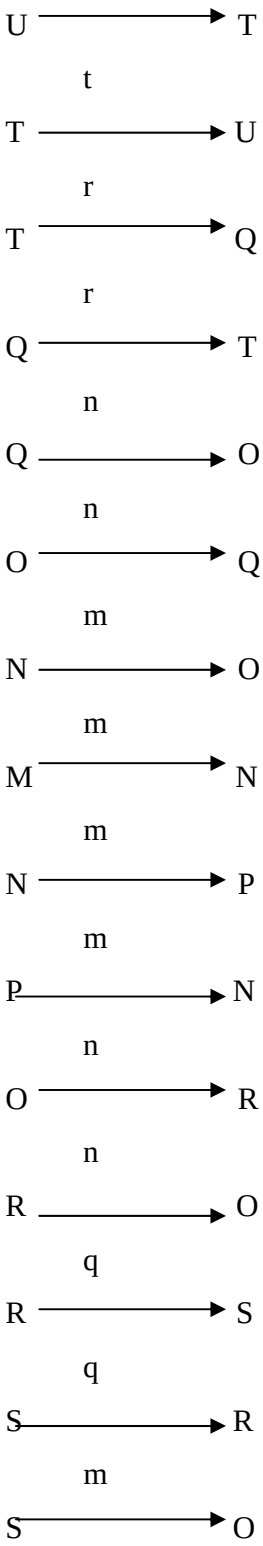
v

V ⟶ U

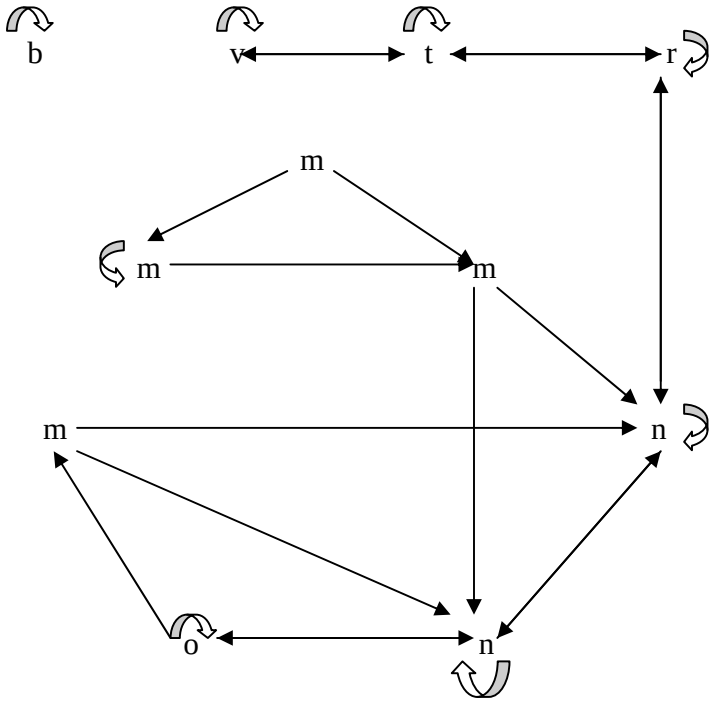
v

U ⟶ V

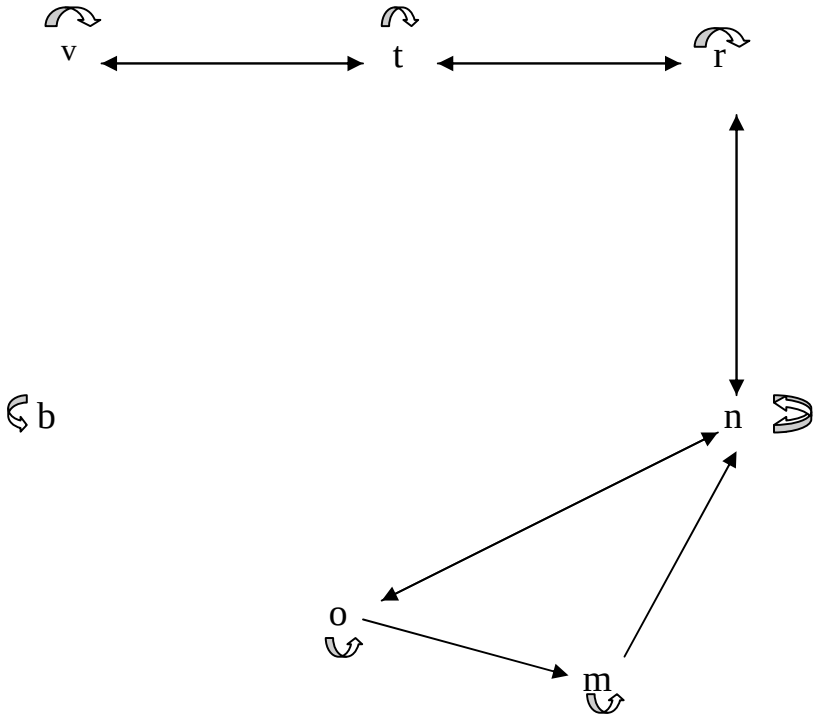
t


U
T
t
T
U
r
T
Q
r
Q
T
n
Q
O
n
O
Q
m
N
O
m
M
N
m
N
P
m
P
N
n
O
R
n
R
O
q
R
S
q
S
R
m
S
O

3. In next step a graph is constructed by taking the EC numbers as nodes and applying the rules described before.

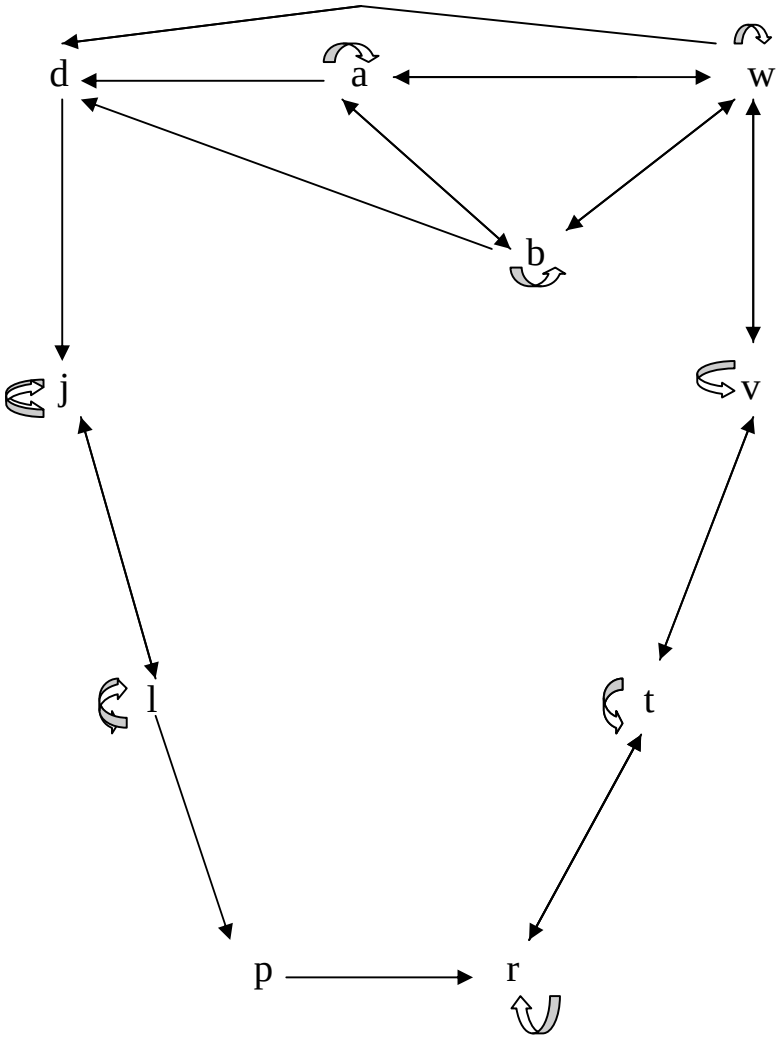


4. Fourth and final step involves simplification of the graph obtained in 3rd step.

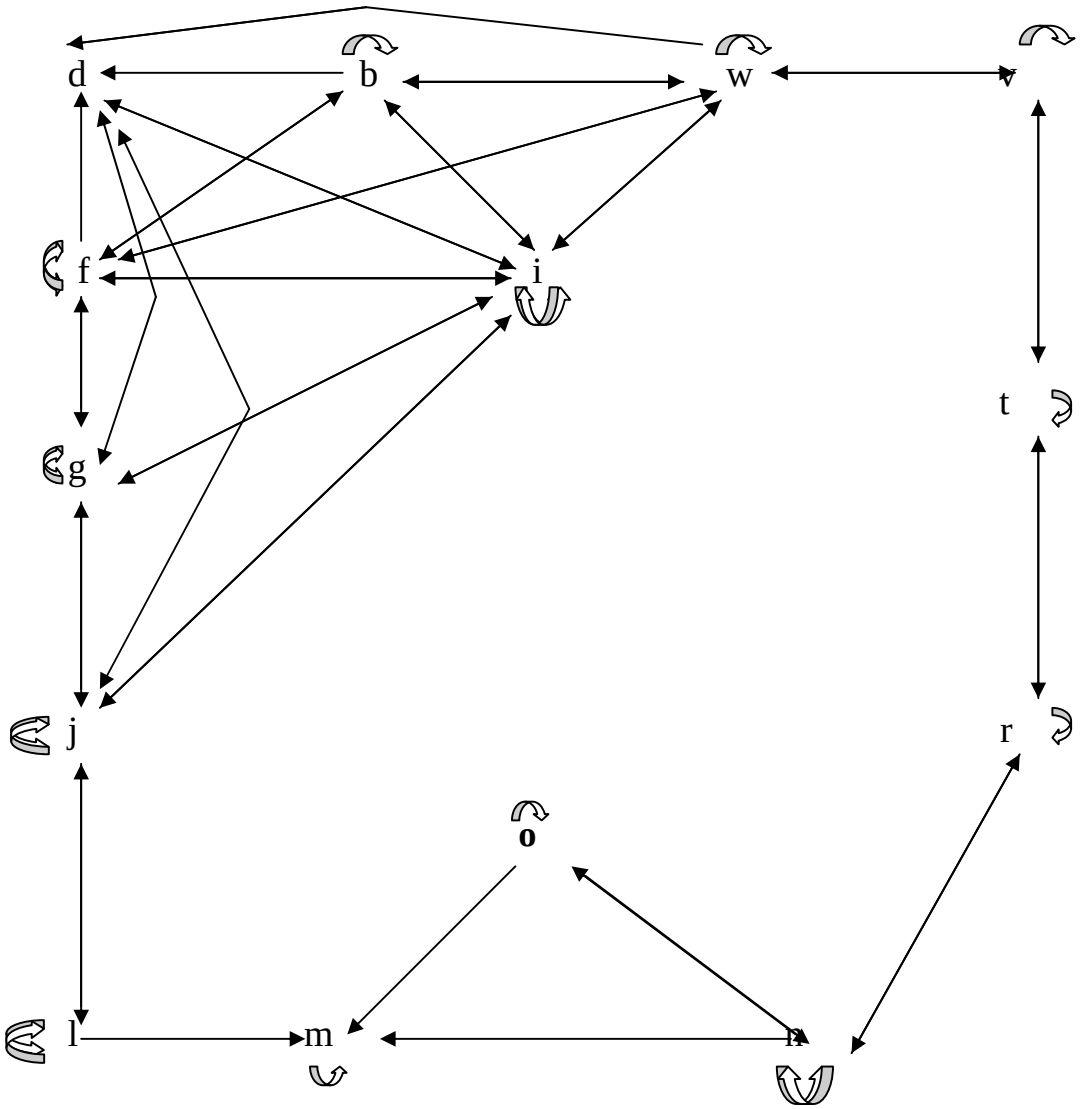


Likewise the graphs all seven species were calculated.

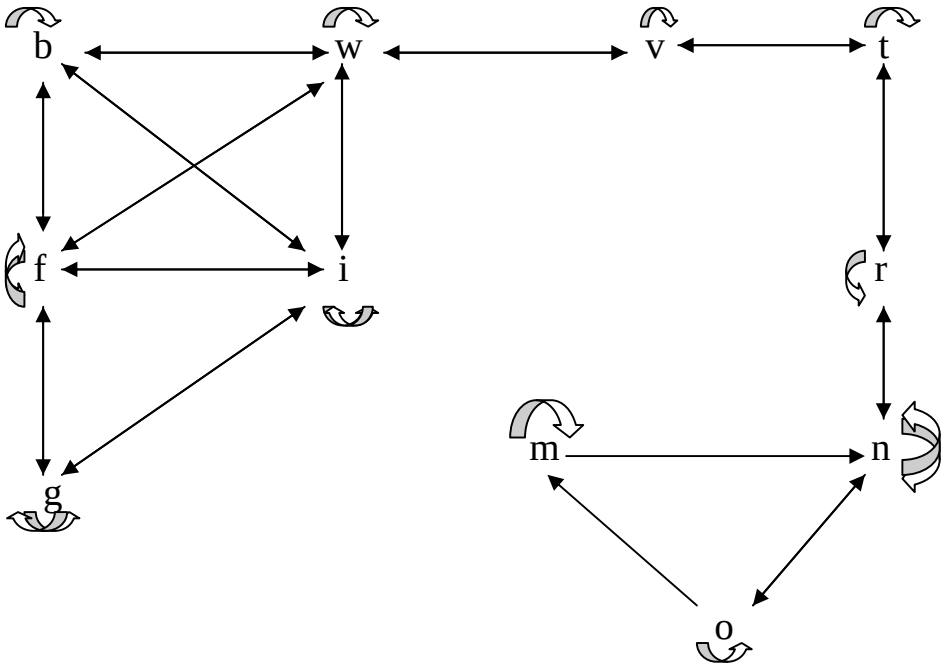
Enzyme Graph of *C. jejuni*:



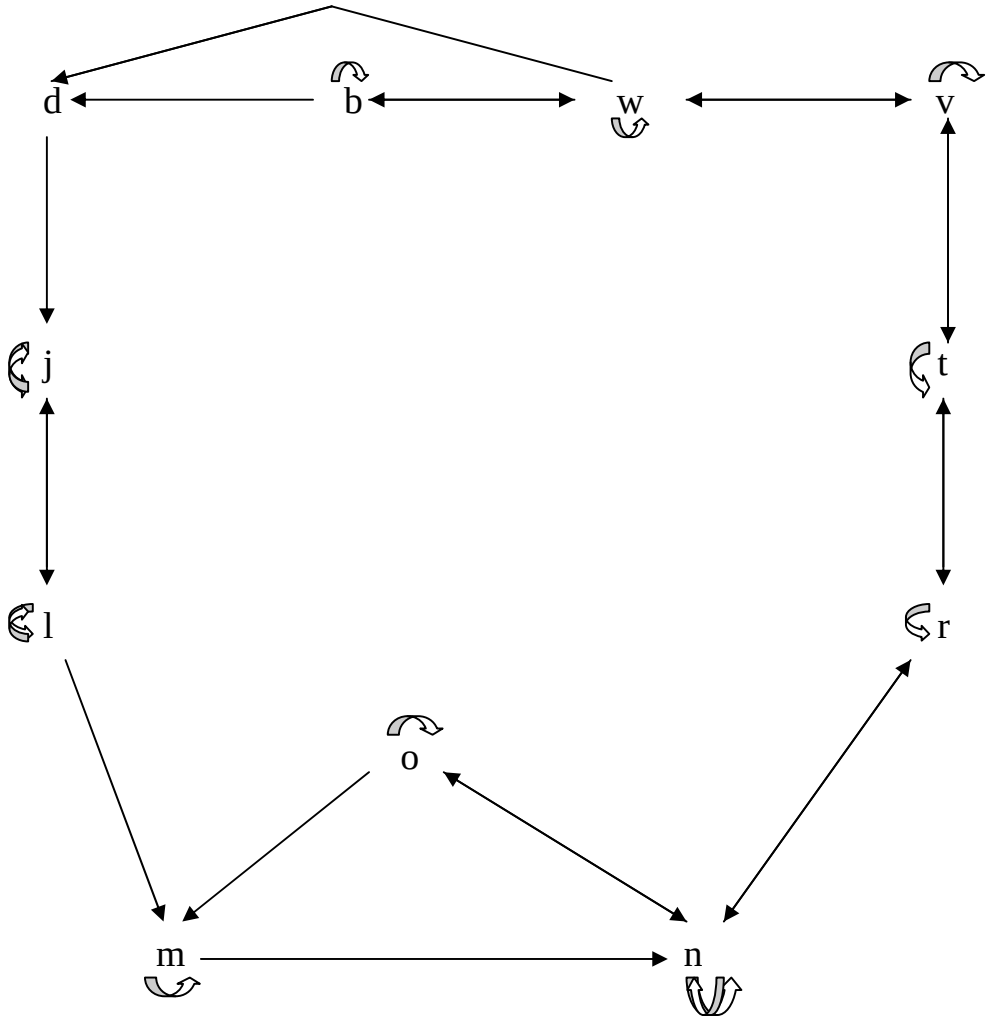
Enzyme graph of *E.coli*:



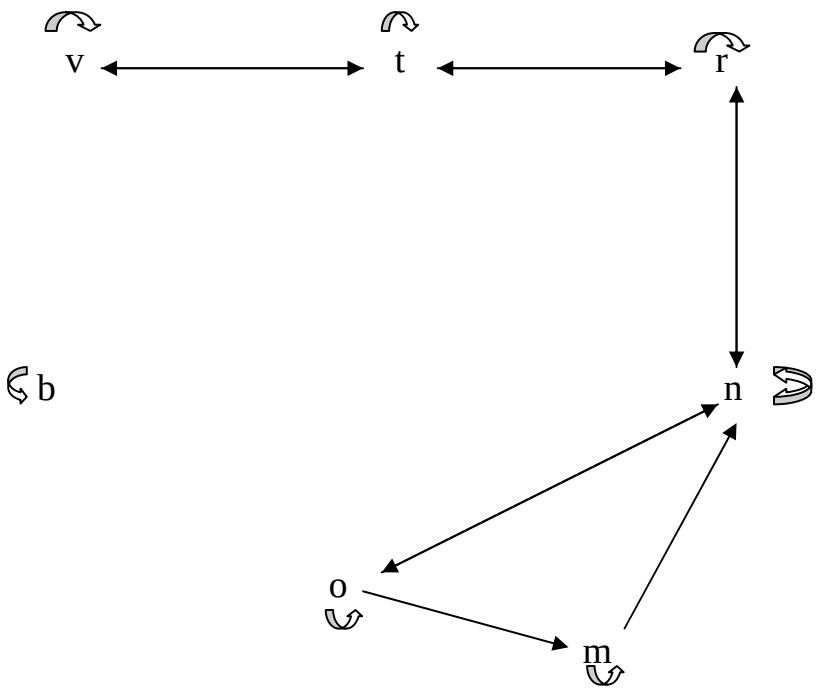
Enzyme Graph of *H. influenza*:



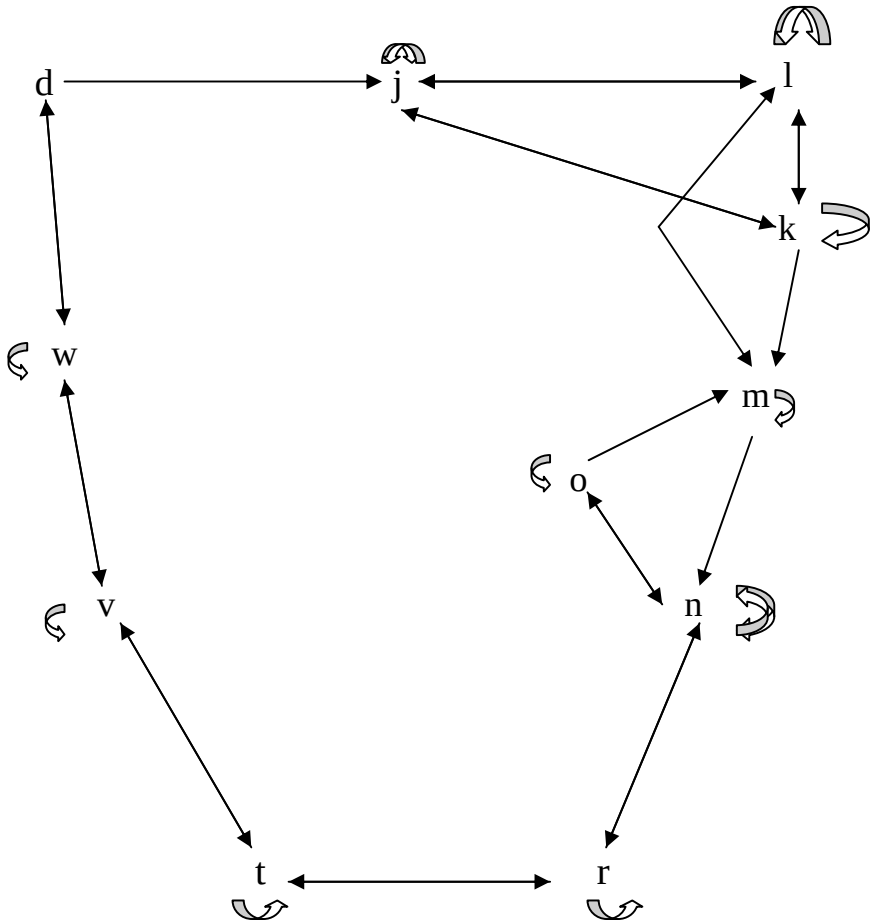
Enzyme Graph of *P. luminescence*:



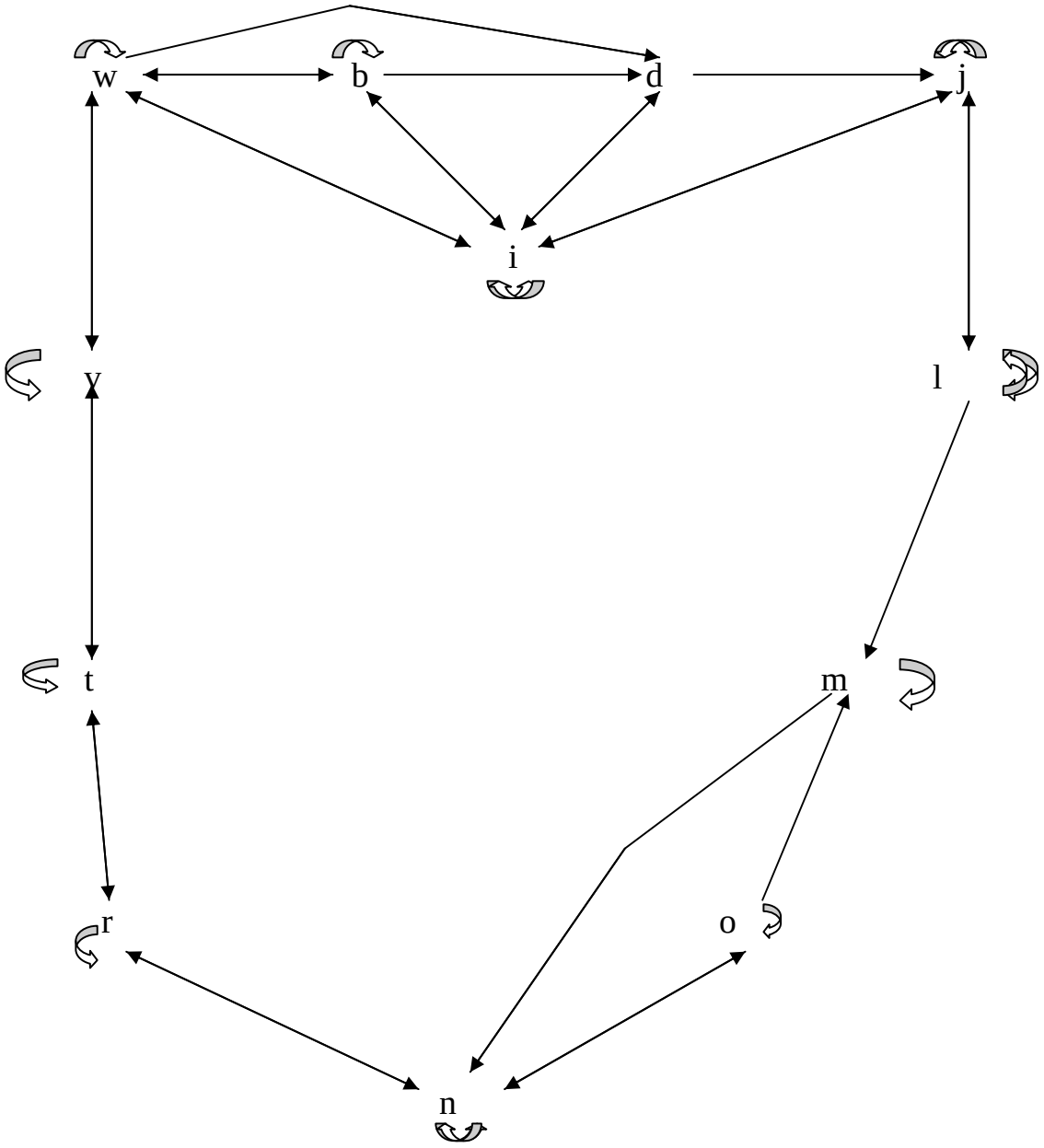
Enzyme Grapf of *W. brevipalpis*:



Enzyme Graph of *X. campestris*:



Enzyme Graph of *Y. pestis*:



VII. CONVERSION OF GRAPHS TO ADJACENCY MATRICES

If graph G has n vertices, then the adjacency matrix is a (n x n) matrix 'A' defined by

$$A(x, y) = \begin{cases} 1 \text{ if } i \longrightarrow j \text{ in } G, \\ 0 \quad \text{othrewise.} \end{cases}$$

x- indicates rows,

y- indicate columns,

Here a to z alphabets are taken as abbreviation of nodes which are actually enzyme commission numbers.

Adjacency matrix representing TCA Cycle enzyme graph of *C. jejuni*:

→ y / x ↓	a	b	d	j	l	p	r	t	v	w
a	1	1	1	0	0	0	0	0	0	1
b	1	1	1	0	0	0	0	0	0	1
d	0	0	0	1	0	0	0	0	0	0
j	0	0	0	2	1	0	0	0	0	0
l	0	0	0	1	2	1	0	0	0	0
p	0	0	0	0	0	0	1	0	0	0
r	0	0	0	0	0	0	1	1	0	0
t	0	0	0	0	0	0	1	1	1	0
v	0	0	0	0	0	0	0	1	1	1
w	1	1	1	0	0	0	0	0	1	1

Adjacency matrix representing TCA Cycle enzyme graph of *E. coli*:

y / x	b	d	f	g	i	j	l	m	n	o	r	t	v	w
b	1	1	1	0	1	0	0	0	0	0	0	0	0	1
d	0	0	0	1	1	1	0	0	0	0	0	0	0	0
f	1	1	2	1	1	0	0	0	0	0	0	0	0	1
g	0	1	1	2	1	1	0	0	0	0	0	0	0	0
i	1	1	1	1	2	1	0	0	0	0	0	0	0	1
j	0	1	0	1	1	2	1	0	0	0	0	0	0	0
l	0	0	0	0	0	1	2	1	0	0	0	0	0	0

m	0	0	0	0	0	0	0	1	0	0	0	0	0	0
n	0	0	0	0	0	0	0	1	2	1	1	0	0	0
o	0	0	0	0	0	0	0	1	1	1	0	0	0	0
r	0	0	0	0	0	0	0	0	1	0	1	1	0	0
t	0	0	0	0	0	0	0	0	0	0	1	1	1	0
v	0	0	0	0	0	0	0	0	0	0	0	1	1	1
w	1	1	1	0	1	0	0	0	0	0	0	0	1	1

Adjacency matrix representing TCA Cycle enzyme graph of *H. influenzae*:

x \ y	b	f	g	i	m	n	o	r	t	v	w
b	1	1	0	1	0	0	0	0	0	0	1
f	1	2	1	1	0	0	0	0	0	0	1
g	0	1	2	1	0	0	0	0	0	0	0
i	1	1	1	2	0	0	0	0	0	0	1
m	0	0	0	0	1	1	0	0	0	0	0
n	0	0	0	0	0	2	1	1	0	0	0
o	0	0	0	0	1	1	1	0	0	0	0
r	0	0	0	0	0	1	0	1	1	0	0
t	0	0	0	0	0	0	0	1	1	1	0
v	0	0	0	0	0	0	0	0	1	1	1
w	1	1	0	1	0	0	0	0	0	1	1

Adjacency matrix representing TCA Cycle enzyme graph of *P. luminescence*:

x \ y	b	d	j	l	m	n	o	r	t	v	w
b	1	1	0	0	0	0	0	0	0	0	1
d	0	0	1	0	0	0	0	0	0	0	0
j	0	0	2	1	0	0	0	0	0	0	0
l	0	0	1	2	1	0	0	0	0	0	0
m	0	0	0	0	1	1	0	0	0	0	0
n	0	0	0	0	0	2	1	1	0	0	0
o	0	0	0	0	1	1	1	0	0	0	0
r	0	0	0	0	0	1	0	1	1	0	0
t	0	0	0	0	0	0	0	1	1	1	0
v	0	0	0	0	0	0	0	0	1	1	1
w	1	1	0	0	0	0	0	0	0	1	1

Adjacency matrix representing TCA Cycle enzyme graph of *W. brevipalpis*:

x \ y	b	m	n	o	r	t	v
b	1	0	0	0	0	0	0
m	0	1	1	0	0	0	0
n	0	0	2	1	1	0	0
o	0	1	1	1	0	0	0
r	0	0	1	0	1	1	0
t	0	0	0	0	1	1	1
v	0	0	0	0	0	1	1

Adjacency matrix representing TCA Cycle enzyme graph of *X. campestris*:

x \ y	d	j	k	l	m	n	o	r	t	v	w
d	0	1	0	0	0	0	0	0	0	0	1
j	0	2	1	1	0	0	0	0	0	0	0
k	0	1	1	1	1	0	0	0	0	0	0
l	0	1	1	2	1	0	0	0	0	0	0
m	0	0	0	1	1	1	0	0	0	0	0
n	0	0	0	0	0	2	1	1	0	0	0
o	0	0	0	0	1	1	1	0	0	0	0
r	0	0	0	0	0	1	0	1	1	0	0
t	0	0	0	0	0	0	0	1	1	1	0
v	0	0	0	0	0	0	0	0	1	1	1
w	1	0	0	0	0	0	0	0	0	1	1

Adjacency matrix representing TCA Cycle enzyme graph of *Y. pestis*:

x \ y	b	d	i	j	l	m	n	o	r	t	v	w
b	1	1	1	0	0	0	0	0	0	0	0	1
d	0	0	1	1	0	0	0	0	0	0	0	0
i	1	1	2	1	0	0	0	0	0	0	0	1
j	0	0	1	2	1	0	0	0	0	0	0	0

l	0	0	0	1	2	1	0	0	0	0	0	0
m	0	0	0	0	0	1	1	0	0	0	0	0
n	0	0	0	0	0	0	2	1	1	0	0	0
o	0	0	0	0	0	1	1	1	0	0	0	0
r	0	0	0	0	0	0	0	1	1	1	0	0
t	0	0	0	0	0	0	0	0	1	1	1	0
v	0	0	0	0	0	0	0	0	0	1	1	1
w	1	1	1	0	0	0	0	0	0	0	1	1

VIII. THE ALOGORITHM COMPUTING SIMILARITY BETWEEN GRAPHS

The algorithm for computing the similarity between two graphs $G1$ and $G2$ is divided in four phases. In the first phase, the similarity score between every pair of nodes *(a, b)* where $a \in G1$ and $b \in G2$ is computed by an iterative process. In the second phase, we construct a bipartite graph using the similarity scores and find a maximal weight matching of this bipartite graph. In the third phase, a similarity measure between every pair of matched nodes is recomputed. Finally, a similarity score between the two graphs is computed by summing the similarity of the matched nodes and by normalizing this sum.

Obtaining similarity scores between nodes

Let $G1 = (V1, E1, ë1)$ where $|V1| = n1$ and $G2 = (V2, E2, ë2)$ where $|V2| = n2$ be two directed graphs. $G1$ and $G2$ are represented by their adjacency matrix $A1$ ($n1 \times n1$) and $A2$ ($n2 \times n2$). A $n1 \times n2$ similarity matrix S, where the entry $S(a, b)$ expresses the similarity between the node $a \in G1$ and node $b \in G2$, is obtained as the limit of a converging iterative process. The similarities between every pair of nodes *(a, b)* where $a \in G1$ and $b \in G2$ are computed simultaneously. We first define a similarity score *Sim* between every pair of objects represented by the nodes of the two graphs. In the case of the enzyme graphs the similarity between enzymes can be defined in a number of ways viz. identity mapping, sequence similarity or structural similarity. The similarity between every pair

of nodes *(a, b)* of $G1$ and $G2$ is then defined by combining the notion of similarity between the objects the nodes represent and the similarity of their neighborhood. The basic intuition behind the approach is that two nodes are similar if they reference and are referenced by similar nodes. A similar approach has been recently developed independently by Melnik *et al.* (2002), Jeh and Widom (2002) and Blondel and Van Dooren (2002). However, they didn't define a similarity score between graphs as the authors did.

The similarity scores between nodes $S(a, b)$ are initialized with $Sim(a, b)$ and then updated simultaneously according to the following mutually recursive rule: two nodes are similar if they link to similar nodes, are referenced by similar nodes, have both missing ingoing (outgoing) edges from (to) similar nodes and have mismatches between edges from (to) dissimilar nodes. The similarity between two nodes *(a, b)* is computed by summing their similarities and subtracting their dissimilarities. The former consists of four terms $A1$–$A4$ and the latter consists of four terms $D1$–$D4$. The first four terms represent the similarity between the presence and absence of edges from and to similar nodes while the remaining four terms represent the mismatches between these edges.

Term $A1(a, b)$ represents the average similarity between the in-neighbors of a (nodes from which a has incoming edges) and the in-neighbors of b. We first obtain the sum of similarities of the pair of nodes $(a2, b2)$ (with $a2 \in G1$ and $b2 \in G2$) from which, a and b have incoming edges. We normalize the sum by dividing it by the total number of in-neighbor pairs, $degin(a).degin(b)$ (degin(a) denotes the number of incoming edges to node a.). A slight technicality here is that either a and /or b may not have any in-neighbors. If both a and b have an in-degree of 0, then the term $A1$ is defined as the sum of similarities of every pair $(a2, b2)$ (with $a2 \in G1$ and $b2 \in G2$) normalized by the total number of such pairs $n1 \times n2$. If only one of them has an in-degree of 0 then $A1$ is set to 0.

Term $A2\ (a, b)$ represents the average similarity between the out-neighbors of a (nodes to which a has outgoing edges) and the out-neighbors of b. It is computed over the pair of nodes $(a2, b2)$ (with $a2 \in G1$ and $b2 \in G2$) to which the nodes a and b have outgoing edges. It is defined analogously to $A1$. The next two terms are motivated by the fact that the absence of edges to similar nodes may be as meaningful as the presence of edges to similar nodes. Term $A3\ (a, b)$ is similar to $A1(a, b)$ except that it works on the complement of the input graphs. It represents the average similarity between the non-in-neighbors of a (nodes from which a has no incoming edges) and the non-in-neighbors of b. We first obtain the sum of similarities of the pair of nodes $(a2, b2)$ (with $a2 \in G1$ and $b2 \in G2$) from which the nodes a and b have no incoming edges. The sum is normalized by dividing by the total number of non-in-neighbor pairs, $(n1 - degin(a)).(n2 - degin(b))$.

Term $A4\ (a, b)$ represents the average similarity between the non-out-neighbors of a (nodes to which a has no outgoing edges) and the non-out-neighbors of b. It is computed over the pair of nodes $(a2, b2)$ (with $a2 \in G1$ and $b2 \in G2$) to which the nodes a and b have no outgoing edges. It is defined analogously to $A3$.

Term $D1\ (a, b)$ represents the dissimilarity between nodes a and b on account of the incoming edges. It is computed over the pair of nodes $(a2, b2)$ (with $a2 \in G1$ and $b2 \in G2$) from which node a has an incoming edges ($a2 \longrightarrow a$) but b does not ($b2 \longrightarrow b$).

Term $D2\ (a, b)$ is the analogue of $D1\ (a, b)$. It considers the similarity of nodes from which a has no incoming edges but b does. Term $D3\ (a, b)$ considers the similarity of nodes to which a has an outgoing edge but b does not. Term $D4\ (a, b)$ is the analogue of $D3$. It considers the similarity of nodes to which a has no outgoing edges but b does.

The similarity scores $S\ (a, b)$ are computed by iteration to a fixed point. We initialize the scores $S0\ (a, b)$ to $Sim\ (a, b)$. The scores $S(k+1)(a, b)$ are then recursively computed

based on *Sk* . Since we are only interested in the relative scores, the scores are normalized after each iteration. Here is the outline of the iterative process.

Initialization:

$S^0(a, b) = Sim(a, b)$ (1)

Iterative step:

$$S^{(k+1)}(a, b) = ((A^k_1(a, b) + A^k_2(a, b) + A^k_3(a, b) + A^k_4(a, b) - D^k_1(a, b) + D^k_2(a, b) + D^k_3(a, b) + D^k_4(a, b)) / 4) \times Sim(a, b) \quad (2)$$

Normalization:

$$S \leftarrow \frac{S}{\| S \|_2} \quad (3)$$

In equation (2), the similarity scores *S (a, b)* are multiplied by *Sim (a, b)* in order to combine the neighborhood similarity with the similarity of the objects represented by the nodes. Since each of the four terms *A*1 to *A*4, and each of the four terms *D*1 to *D*4 have a range between -1 and 1, *S(a, b)* is also divided by 4 in order to have a range between -1 and 1. From the equations, we see that the similarity scores are symmetric, i.e. *S(a, b)* = *S(b, a)*. The convergence of the above iterative process is considered in (Heymens and Singh, 2002).

Bipartite graph matching

At the end of the first phase, we obtain a matrix *S* that captures the similarity between every pair of nodes of the input graph. The second phase uses these similarities to find the best matching between the graphs. In order to achieve this, we build a bipartite graph and execute a bipartite graph matching algorithm. Given vertex sets *V*1 of *G*1 and *V*2 of *G*2, we construct a bipartite graph *G = (V*1, *V*2, *S)* where *S* is the similarity matrix obtained during the first phase. Once this bipartite graph has built, we find the best matching of the graph using an *O((n*1 + *n*2)3*)* Hungarian algorithm (Hopcroft and Karp,

1973; Kuhn, 1955; Baier Saip and Lucchesi, 1993). With the best matching so obtained, we define an $n1 \times n2$ boolean matrix M whose entry $M(a, b)$ is set to 1 if nodes a and b have been matched.

Computation of similarity scores between matched nodes

After we find the best correspondence between graphs $G1$ and $G2$, we need to obtain the similarity score for this correspondence. As in the first phase, we combine the structural similarity with the node similarity to compute this score. We perform one iteration of a system of equations similar to $A1$–$A4$ and $D1$–$D4$. The new set of equations A'_1–A'_4 and D'_1–D'_4 is similar to the previous (unprimed) one except that we use $M(a, b)$ instead of $Sim(a, b)$. We also use a new normalization that is square root of the previous one. This is necessary since the maximum size of a matching is the smaller of the input graph sizes; specifically, if a graph is compared to itself then $M(a, b)$ is given by the identity mapping: the similarity terms A'_1–A'_4 reduce to 1 and the dissimilarity terms D'_1 –D'_4 reduce to 0.

Terms A'_1–A'_4 and D'_1 –D'_4 incorporate the similarity and the dissimilarity of the best match between graphs $G1$ and $G2$. We combine these terms and multiply by the similarity of the nodes to obtain the final value of $S(a, b)$.

Computing graph similarity score

Finally, to obtain the similarity score $SG1.G2$ between the graphs $G1$ and $G2$, we sum the similarity scores computed in the previous phase over the pair of matched nodes, and normalize the sum by the square root of the product of the number of nodes of $G1$ and $G2$, in order to have a similarity score between -1 and 1. When $G1 = G2$, the similarity score will be equal to 1.

$$SG1,G2 = \frac{\sum_{a \in G1,\, b \in G2, M(a,b)=1} S(a,b)}{\sqrt{n1.n2}} \qquad (4)$$

IX. IMPLEMENTATION OF ALGORITHM IN C CODE

Input- Two adjacency matrices corresponding to TCA Cycle metabolic maps of *Campylobacter jejuni* and *Escherichia coli* taken from KEGG.

Tips to understand the code:

count- integer type variable that indicates iteration number

z- globally declared pointer of FILE type

dv- float type variable that stores difference value of a particular shell between two successive iterations

sdv- float type variable, that stores sum of different values of all shells between two successive iterations

asdv- float type variable that stores average of sum of different values of all shells between two successive iterations

s1, s2, s3, s4, p, data- locally declared float type variables used in different user defined functions throughout the code

sim1, sim2, sim3, sim4- float type variables used to store similarity values during comparison of two EC numbers at respective steps in match
function

sum- float type variable used to store the sum of sim1, sim2, sim3, sim4 and finally this value is returned by match().

The Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
```

```
#define n1 10
#define n2 14
float So[n1][n2];
FILE *z;
int ind1(int),ind2(int),outd1(int),outd2(int);
floa match(char[],char[]),sum(),sim(int,int),simv(int,int),assignA1(int,int),assignA2
(int,int),assignA3(int,int),assignA4(int,int),assignD1(int,int),assignD2(int,int),assignD3
(int,int),assignD4(int,int);
int Cj[n1][n1]={{1,1,1,0,0,0,0,0,0,1},
                {1,1,1,0,0,0,0,0,0,1},
                {0,0,0,1,0,0,0,0,0,0},
                {0,0,0,2,1,0,0,0,0,0},
                {0,0,0,1,2,1,0,0,0,0},
                {0,0,0,0,0,0,1,0,0,0},
                {0,0,0,0,0,0,1,1,0,0},
                {0,0,0,0,0,0,1,1,1,0},
                {0,0,0,0,0,0,0,1,1,1},
                {1,1,1,0,0,0,0,0,1,1}};

int Ec[n2][n2]={{1,1,1,0,1,0,0,0,0,0,0,0,0,1},
                {0,0,0,1,1,1,0,0,0,0,0,0,0,0},
                {1,1,2,1,1,0,0,0,0,0,0,0,0,1},
                {0,1,1,2,1,1,0,0,0,0,0,0,0,0},
                {1,1,1,1,2,1,0,0,0,0,0,0,0,1},
                {0,1,0,1,1,2,1,0,0,0,0,0,0,0},
                {0,0,0,0,0,1,2,1,0,0,0,0,0,0},
                {0,0,0,0,0,0,0,1,0,0,0,0,0,0},
                {0,0,0,0,0,0,0,1,2,1,1,0,0,0},
                {0,0,0,0,0,0,0,1,1,1,0,0,0,0},
                {0,0,0,0,0,0,0,0,1,0,1,1,0,0},
                {0,0,0,0,0,0,0,0,0,0,1,1,1,0},
                {0,0,0,0,0,0,0,0,0,0,0,1,1,1},
                {1,1,1,0,1,0,0,0,0,0,0,0,1,1}};
```

// Structure declaration

```
struct node
                {
                        char EC[11];
                };
typedef struct node NODE;
NODE E1[n1]={"6.4.1.1","4.1.1.49","2.3.3.1","4.2.1.3","1.1.1.42","1.2.7.3","6.2.1.5",
             "1.3.99.1","4.2.1.2","1.1.1.37"};
NODE E2[n2]={"4.1.1.49","2.3.3.1","4.1.3.34","2.8.3.10","4.1.3.6","4.2.1.3","1.1.1.42",
```

```
"1.2.4.2","2.3.1.61","1.8.1.4","6.2.1.5","1.3.99.1","4.2.1.2","1.1.1.37"};
main()
{
 int x=0,y=0,count=0;
 float S1[n1][n2],Sim[n1][n2],S[25][n1][n2];
 z=fopen("opt","w");
 for(x=0;x<n1;x++)
   {
    for(y=0;y<n2;y++)
      {
       Sim[x][y]=match(E1[x].EC,E2[y].EC);
       So[x][y]=Sim[x][y];
       fprintf(z,"%f\t",So[x][y]);
      }
   }
 fclose(z);
 for(count=0;count<25;count++)
   {
    float m=0.000;
    for(x=0;x<n1;x++)
      {
       for(y=0;y<n2;y++)
            {
             S1[x][y]=(((assignA1(x,y)+assignA2(x,y)+assignA3(x,y)+assignA4(x,y)-
                        assignD1(x,y)-assignD2(x,y)-assignD3(x,y)-
                        assignD4(x,y))/4)*sim(x,y));
             m=m+(S1[x][y]*S1[x][y]);
            }
         }
    for(x=0;x<n1;x++)
      {
       for(y=0;y<n2;y++)
         {
          S1[x][y]=S1[x][y]/sqrt(m);
          S[count][x][y]=S1[x][y];
         }
      }
    if(count<1)
      {
       z=fopen("opt","w");
       for(x=0;x<n1;x++)
         {
          for(y=0;y<n2;y++)
           {
```

```
            fprintf(z,"%f\t",S[count][x][y]);
           }
       }
     fclose(z);
    }
if(count>=1)
  {
    float dv=0.000,sdv=0.000,asdv=0.000;
    for(x=0;x<n1;x++)
       {
        for(y=0;y<n2;y++)
           {
            dv=S[count][x][y]-S[count-1][x][y];
            dv=fabs(dv);
            sdv=sdv+dv;
           }
       }
    asdv=sdv/(n1*n2);
    printf("average sum of difference values of %d iteraration: %f\n",count,asdv);
    if(asdv>0.00009)
      {
       z=fopen("opt","w");
       for(x=0;x<n1;x++)
          {
           for(y=0;y<n2;y++)
              {
               fprintf(z,"%f\t",S[count][x][y]);
              }
           }
       fclose(z);
      }
    else
      {
       printf("the similarity values are almost converged.\n");
       printf("result after %d iterations:\n",count);
       for(x=0;x<n1;x++)
          {
           for(y=0;y<n2;y++)
              {
               printf("%f,",S[count-1][x][y]);
              }
           printf("\n");
          }
       exit;
```

```
        }
      }
    }
 }
```

// Given the serial no. of node in Graph1, ind1() returns no. of indegree of the node.

```
int ind1(int p)
{
 int i=0,x=0,y=0;
 for(x=p;x<n1;x++)
    {
      for(y=0;y<n1;y++)
          {
            i=i+Cj[y][x];
          }
      break;
    }
 return (i);
}
```

//Given the serial no. of node in Graph2, ind2() returns no. of indegree of the node.

```
int ind2(int p)
{
 int i=0,x=0,y=0;
 for(x=p;x<n2;x++)
    {
      for(y=0;y<n2;y++)
          {
            i=i+Ec[y][x];
          }
    break;
   }
 return (i);
}
```

//Given the serial no. of node in Graph1, outd1() returns no. of outdegree of the node.

```
int outd1(int p)
{
 int i=0,x=0,y=0;
 for(x=p;x<n1;x++)
```

```
    {
     for(y=0;y<n1;y++)
          {
           i=i+Cj[x][y];
          }
     break;
    }
 return (i);
}
```

//Given the serial no. of node in Graph2 , outd2() returns no. of outdegree of the node.

```
int outd2(int p)
{
 int i=0,x=0,y=0;
 for(x=p;x<n2;x++)
    {
     for(y=0;y<n2;y++)
        {
          i=i+Ec[x][y];
        }
    break;
   }
 return (i);
}
```

//Given the row and column value of a shell, assignA1() returns A1 value corresponding to the shell.

```
float assignA1(int p, int q)
{
 int m,n;
 float A1[n1][n2];
 if((ind1(p)==0) && (ind2(q)==0))
 A1[p][q]=sum()/(n1*n2);
 if((ind1(p)==0) || (ind2(q)==0))
 A1[p][q]=0.000;
 if((ind1(p)!=0) && (ind2(q)!=0))
            {
             float s1=0.000;
             for(m=0;m<n1;m++)
               {
                 if(Cj[m][p]!=0)
```

```
                {
                 for(n=0;n<n2;n++)
                    {
                     if(Ec[n][q]!=0)
                       {
                        s1=s1+simv(m,n);
                       }
                     }
                 }
               }
            A1[p][q]=s1/(ind1(p)*ind2(q));
          }
 return(A1[p][q]);
}
```

//Given the row and column value of a shell, assignA2() returns A2 value corresponding to the shell.

```
float assignA2(int p,int q)
{
 int m,n;
 float A2[n1][n2];
 if((outd1(p)==0) && (outd2(q)==0))
 A2[p][q]=sum()/(n1*n2);
 if((outd1(p)==0) || (outd2(q)==0))
 A2[p][q]=0.000;
 if((outd1(p)!=0) && (outd2(q)!=0))
   {
    float s2=0.000;
    for(m=0;m<n1;m++)
       {
        if(Cj[p][m]!=0)
          {
           for(n=0;n<n2;n++)
              {
               if(Ec[q][n]!=0)
                 {
                  s2=s2+simv(m,n);
                 }
              }
          }
       }
   A2[p][q]=s2/(outd1(p)*outd2(q));
  }
```

```
 return(A2[p][q]);
}
```

//Given the row and column value of a shell, assignA3() returns A3 value corresponding to the shell.

```
float assignA3(int p,int q)
{
  int m,n;
  float A3[n1][n2];
  if(((n1-ind1(p))==0) && ((n2-ind2(q))==0))
 A3[p][q]=sum()/(n1*n2);
 if(((n1-ind1(p))==0) || ((n1-ind2(q))==0))
 A3[p][q]=0.000;
 if((ind1(p)!=n1) && (ind2(q)!=n2))
   {
     float s3=0.000;
     for(m=0;m<n1;m++)
        {
          if(Cj[m][p]==0)
           {
            for(n=0;n<n2;n++)
               {
                 if(Ec[n][q]==0)
                   {
                     s3=s3+simv(m,n);
                   }
               }
           }
    }
   A3[p][q]=s3/((n1-ind1(p)) * (n2-ind2(q)));
  }
 return(A3[p][q]);
}
```

//Given the row and column value of a shell, assignA4() returns A4 value corresponding to the shell.

```
float assignA4(int p,int q)
{
  int m,n;
  float A4[n1][n2];
  if(((n1-outd1(p))==0) && ((n2-outd2(q))==0))
  A4[p][q]=sum()/(n1*n2);
```

```
 if(((n1-outd1(p))==0) || (n2-outd2(q)==0))
 A4[p][q]=0.000;
 if((outd1(p)!=n1) && (outd2(q)!=n2))
   {
    float s4=0.000;
    for(m=0;m<n1;m++)
       {
        if(Cj[p][m]==0)
          {
           for(n=0;n<n2;n++)
              {
               if(Ec[q][n]==0)
                 {
                  s4=s4+simv(m,n);
                 }
              }
          }
       }
      A4[p][q]=s4/((n1-outd1(p)) * (n2-outd2(q)));
    }
 return(A4[p][q]);
}
```

//Given the row and column value of a shell, assignD1() returns D1 value corresponding to the shell.

```
float assignD1(int p,int q)
{
 int m,n;
 float D1[n1][n2];
 if((ind1(p)==0) && ((n2-ind2(q))==0))
 D1[p][q]=sum()/(n1*n2);
 if((ind1(p)==0) || ((n2-ind2(q))==0))
 D1[p][q]=0.000;
 if((ind1(p)!=0) && (ind2(q)!=n2))
   {
    float s1=0.000;
    for(m=0;m<n1;m++)
       {
        if(Cj[m][p]!=0)
          {
           for(n=0;n<n2;n++)
              {
               if(Ec[n][q]==0)
```

```
                    {
                      s1=s1+simv(m,n);
                    }
                  }
              }
         }
      D1[p][q]=s1/(ind1(p) * (n2-ind2(q)));
    }
 return(D1[p][q]);
}
```

//Given the row and column value of a shell, assignD2() returns D2 value corresponding to the shell.

```
float assignD2(int p,int q)
{
  int m,n;
  float D2[n1][n2];
  if(((n1-ind1(p))==0) && (ind2(q)==0))
  D2[p][q]=sum()/(n1*n2);
  if(((n1-ind1(p))==0) || (ind2(q)==0))
  D2[p][q]=0.000;
  if((ind1(p)!=n1) && (ind2(q)!=0))
    {
      float s2=0.000;
      for(m=0;m<n1;m++)
         {
          if(Cj[m][p]==0)
            {
              for(n=0;n<n2;n++)
                 {
                   if(Ec[n][q]!=0)
                     {
                       s2=s2+simv(m,n);
                     }
                  }
            }
         }
       D2[p][q]=s2/((n1-ind1(p))*ind2(q));
      }
 return(D2[p][q]);
}
```

//Given the row and column value of a shell, assignD3() returns D3 value corresponding to the shell.

```
float assignD3(int p,int q)
{
  int m,n;
  float D3[n1][n2];
  if((outd1(p)==0) && ((n2-outd2(q))==0))
  D3[p][q]=sum()/(n1*n2);
  if((outd1(p)==0) || ((n2-outd2(q))==0))
  D3[p][q]=0.000;
  if((outd1(p)!=0) && (outd2(q)!=n2))
    {
     float s3=0.000;
     for(m=0;m<n1;m++)
        {
          if(Cj[p][m]!=0)
            {
              for(n=0;n<n2;n++)
                 {
                   if(Ec[q][n]==0)
                     {
                       s3=s3+simv(m,n);
                     }
                 }
            }
        }
      D3[p][q]=s3/(outd1(p) * (n2-outd2(q)));
     }
 return(D3[p][q]);
}
```

//Given the row and column value of a shell, assignD4() returns D4 value corresponding to the shell.

```
float assignD4(int p,int q)
{
  int m,n;
  float D4[n1][n2];
  if(((n1-outd1(p))==0) && (outd2(q)==0))
  D4[p][q]=sum()/(n1*n2);
  if(((n1-outd1(p))==0) || (outd2(q)==0))
  D4[p][q]=0.000;
  if((outd1(p)!=n1) && (outd2(q)!=0))
```

```
   {
    float s4=0.000;
    for(m=0;m<n1;m++)
       {
        if(Cj[p][m]==0)
          {
           for(n=0;n<n2;n++)
              {
               if(Ec[q][n]!=0)
                 {
                  s4=s4+simv(m,n);
                 }
              }
          }
       }
     D4[p][q]=s4/((n1-outd1(p)) * outd2(q));
   }
 return(D4[p][q]);
}
```

//Given the row and column value of a shell, simv() returns similarity value corresponding to the shell by opening file opt. Returned value for a particular shell is different for different iterations.

```
float simv(int p,int q)
{
  int x=0,y=0;
  float So[n1][n2],data;
  z=fopen("opt","r");
  for(x=0;x<n1;x++)
     {
      for(y=0;y<n2;y++)
         {
          fscanf(z,"%f",&data);
          So[x][y]=data;
         }
     }
 fclose(z);
 return(So[p][q]);
}
```

//sum() returns sum of similarity values present in opt file. Returned value is different for different iterations.

```
float sum()
{
  int x=0,y=0;
  float p=0.000,data;
  z=fopen("opt","r");
  for(x=0;x<n1;x++)
      {
       for(y=0;y<n2;y++)
           {
             fscanf(z,"%f",&data);
             So[x][y]=data;
             p=p+So[x][y];
           }
      }
 fclose(z);
 return(p);
}
```

//Given the row and column value of a shell, sim() returns similarity value calculated by comparison of EC numbers, corresponding to the shell. Returned value for a particular shell is same for different iterations.

```
 float sim(int p,int q)
{
  float Sim[n1][n2];
  Sim[p][q]=match(E1[p].EC,E2[q].EC);
  return(Sim[p][q]);
}
```

//Given the EC numbers corresponding to two nodes, match() returns similarity value present between them. Returned value is constant for different iterations.

```
float match(char EC1[11],char EC2[11])
{
 float sum,sim1=0,sim2=0,sim3=0,sim4=0;
 while(EC1[0]!=EC2[0])
             {
              sim1=0;
              goto end;
             }
 while(EC1[0]==EC2[0])
             {
              sim1=0.25;
              break;
```

```
            }
 while((EC1[2]!=EC2[2]) && ((EC1[3]=='.') && (EC2[3]=='.')))
     {
      goto end;
     }
 while((EC1[2]==EC2[2]) && ((EC1[3]=='.')&&(EC2[3]=='.')))
     {
      sim2=0.25;
      break;
     }
 while(((EC1[2]!=EC2[2]) || (EC1[3]!=EC2[3])) && ((EC1[4]=='.')&&(EC2[4]=='.')))
     {
      goto end;
     }
 while(((EC1[2]==EC2[2]) && (EC1[3]==EC2[3]))  && ((EC1[4]=='.') &&
(EC2[4]=='.')))
     {
      sim2=0.25;
      break;
     }
 while((EC1[4]!=EC2[4]) && ((EC1[5]=='.') && (EC2[5]=='.')))
     {
      goto end;
     }
 while((EC1[4]==EC2[4]) && ((EC1[5]=='.') && (EC2[5]=='.')))
     {
      sim3=0.25;
      break;
     }
 while(((EC1[5]!=EC2[5]) && ((EC1[6]=='.') && (EC2[6]=='.'))) && ((EC1[4]=='.') &&
(EC2[4]=='.')))
     {
      goto end;
     }
 while(((EC1[5]==EC2[5]) && ((EC1[6]=='.') && (EC2[6]=='.'))) && ((EC1[4]=='.')
&& (EC2[4]=='.')))
     {
          sim3=0.25;
          break;
     }
 while(((EC1[4]!=EC2[4]) || (EC1[5]!=EC2[5])) && ((EC1[6]=='.')  && (EC2[6]=='.')))
     {
      goto end;
     }
```

```
 while((((EC1[4]==EC2[4]) && (EC1[5]==EC2[5])) && ((EC1[6]=='.') &&
(EC2[6]=='.'))) && ((EC1[4]!='.') && (EC2[4]!='.')))
     {
          sim3=0.25;
          break;
     }
 while(((EC1[5]!=EC2[5]) || (EC1[6]!=EC2[6])) && ((EC1[7]=='.')  && (EC2[7]=='.')))
     {
      goto end;
     }
 while(((EC1[5]==EC2[5]) && (EC1[6]==EC2[6])) && ((EC1[7]=='.') &&
(EC2[7]=='.')))
     {
      sim3=0.25;
      break;
     }
 while((EC1[6]!=EC2[6]) && ((EC1[7]=='\0') && (EC2[7]=='\0')))
     {
      goto end;
     }
 while((EC1[6]==EC2[6]) && ((EC1[7]=='\0') && (EC2[7]=='\0')))
     {
      sim4=0.25;
      break;
     }
 while(((EC1[6]!=EC2[6]) || (EC1[7]!=EC2[7])) && ((EC1[8]=='\0') &&
(EC2[8]=='\0')))
     {
      goto end;
     }
 while((EC1[6]==EC2[6]) && (EC1[7]==EC2[7]) && (EC1[8]=='\0') &&
(EC2[8]=='\0') && (EC1[5]=='.') && (EC2[5]=='.'))
     {
      sim4=0.25;
      break;
     }
 while(((EC1[6]!=EC2[6]) || (EC1[7]!=EC2[7]) || (EC1[8]!=EC2[8])) && ((EC1[9]=='\0')
&& (EC2[9]=='\0')))
     {
      goto end;
     }
 while((EC1[6]==EC2[6]) && (EC1[7]==EC2[7]) && (EC1[8]==EC2[8])  &&
(EC1[9]=='\0') && (EC2[9]=='\0') && (EC1[5]=='.') &&
       (EC2[5]=='.'))
```

```
      {
       sim4=0.25;
       break;
      }
 while((EC1[7]!=EC2[7]) && ((EC1[8]=='\0') && (EC2[8]=='\0')))
      {
       goto end;
      }
 while(((EC1[7]==EC2[7]) && ((EC1[8]=='\0') && (EC2[8]=='\0'))) &&((EC1[6]=='.')
&& (EC2[6]=='.')))
      {
       sim4=0.25;
       break;
      }
 while(((EC1[7]!=EC2[7]) || (EC1[8]!=EC2[8])) && ((EC1[9]=='\0') &&
(EC2[9]=='\0')))
      {
       goto end;
      }
 while((((EC1[7]==EC2[7]) && (EC1[8]==EC2[8])) && ((EC1[9]=='\0') &&
(EC2[9]=='\0'))) && ((EC1[6]=='.') && (EC2[6]=='.')))
      {
       sim4=0.25;
       break;
      }
 while((((EC1[7]!=EC2[7]) || (EC1[8]!=EC2[8])) || (EC1[9]!=EC2[9])) &&
((EC1[10]=='\0') && (EC2[10]=='\0')))
      {
        goto end;
      }
 while(((((EC1[7]==EC2[7]) && (EC1[8]==EC2[8])) && (EC1[9]==EC2[9])) &&
((EC1[10]=='\0') && (EC2[10]=='\0'))) &&((EC1[6]=='.')
          && (EC2[6]=='.')))
      {
       sim4=0.25;
       break;
      }
 while((EC1[8]!=EC2[8]) && ((EC1[9]=='\0') && (EC2[9]=='\0')))
      {
       goto end;
      }
 while(((EC1[8]==EC2[8]) && ((EC1[9]=='\0') && (EC2[9]=='\0'))) && ((EC1[7]=='.')
&& (EC2[7]=='.')))
      {
```

```
      sim4=0.25;
      break;
     }
 while((EC1[8]!=EC2[8]) || (EC1[9]!=EC2[9]) && ((EC1[10]=='\0') &&
(EC2[10]=='\0')))
     {
      goto end;
     }
 while((((EC1[8]==EC2[8]) && (EC1[9]==EC2[9])) && ((EC1[10]=='\0') &&
            (EC2[10]=='\0'))) && ((EC1[7]=='.') && (EC2[7]=='.')))
     {
      sim4=0.25;
      break;
     }
 while(((EC1[8]!=EC2[8]) || (EC1[9]!=EC2[9]) || (EC1[10]!=EC2[10]))
          && ((EC1[11]=='\0') && (EC2[11]=='\0')))
     {
      goto end;
     }
 while((EC1[8]==EC2[8]) && (EC1[9]==EC2[9]) && (EC1[10]==EC2[10]) &&
(EC1[11]=='\0') && (EC2[11]=='\0') && (EC1[7]=='.')
            && (EC2[7]=='.'))
     {
      sim4=0.25;
      break;
     }
  end:
    sum=sim1+sim2+sim3+sim4;
  return(sum);
 }
```

X. Output Values of similarity matrix (itearation wise):

iteration 0:
0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.023858,-0.000000,-0.000000,0.000000,
0.294082,0.000000,0.137321,0.000000,0.089937,0.002352,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.030331,0.000000,
0.000000,0.355627,0.000000,0.036619,0.000000,0.000000,0.000000,-0.000000,-
0.083764,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,
-0.020847,0.000000,-0.011746,0.000000,0.017949,0.161988,0.000000,-0.000000,-
0.000000,0.000000,-0.000000,-0.000000,-0.032042,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.231908,0.030891,-
0.000000,0.006905,-0.000000,-0.027618,0.000000,-0.019640,

-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.016499,0.085298,
0.000000,-0.005035,0.000000,0.025317,-0.000000,-0.041232,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000, 0.000000,0.000000,
0.000000,0.000000,0.467909,0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.017779,-0.029436,
0.000000,-0.017878,0.000000,0.426112, 0.000000,-0.019219,
-0.015134,-0.000000,-0.025914,-0.000000,-0.032889,-0.072221,0.000000,0.000000,-
0.000000,-0.000000,-0.000000,0.000000, 0.392575,0.000000,
0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,-0.052212,-0.047364,-
0.000000,-0.026302,-0.000000,-0.030272,-0.000000,0.293200,

iteration 1:
0.000000,0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.044395,-0.000000,-0.000000,0.000000,
0.291921,0.000000,0.130389,0.000000,0.117926,0.000069,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.009918,0.000000,
0.000000,0.243477,0.000000,0.021275,0.000000,0.000000,0.000000,-0.000000,-
0.011162,0.000000,-0.000000,-0.000000,-0.000000,0.000000,
-0.020572,0.000000,-0.013185,0.000000,0.010577,0.176252,0.000000,0.000000,-
0.000000,0.000000,-0.000000,-0.000000,-0.130003,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,-0.000000,0.000000,0.222991,0.041875,
0.000000,0.025957,-0.000000,-0.046730,-0.000000,-0.086347,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.021073,0.054725,
0.000000,-0.001686,0.000000,0.026364,-0.000000,-0.039355,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.431211,0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.034975,-0.022608,
0.000000,-0.009335,0.000000,0.484179, 0.000000,-0.017714,
-0.005088,-0.000000,-0.005853,-0.000000,-0.020816,-0.115641,-0.000000,-0.000000,-
0.000000,-0.000000,0.000000,0.000000, 0.437072,0.000000,
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.084185,-0.016401,-
0.000000,-0.003020,-0.000000,-0.024212, 0.000000,0.265403,

iteration 2:
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.043201,-0.000000,-0.000000,0.000000,
0.260024,0.000000,0.114744,0.000000,0.106463,-0.001814,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.007307,0.000000,
0.000000,0.242412,0.000000,0.026536,0.000000,0.000000,0.000000,-0.000000,-
0.010174,0.000000,-0.000000,-0.000000,-0.000000,0.000000,
-0.019679,0.000000,-0.011220,0.000000,0.011881,0.185303,0.000000,0.000000,
0.000000,0.000000,-0.000000,-0.000000,-0.170244,-0.000000,

-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.234086,0.038117,
0.000000,0.022428,-0.000000,-0.053519,-0.000000,-0.103255,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.023057,0.057978,
0.000000,-0.000255,0.000000,0.019952,-0.000000,-0.038858,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.403914,0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.040347,-0.025669,
0.000000,-0.016293,0.000000,0.485431, 0.000000,-0.008889,
-0.005154,-0.000000,-0.007246,-0.000000,-0.024330,-0.133263,-0.000000,-0.000000,-
0.000000,-0.000000,0.000000,0.000000, 0.450914,0.000000,
0.000000,0.000000,0.000000,-0.000000,0.000000,-0.000000,-0.096316,-0.015501,-
0.000000,-0.004334,-0.000000,-0.017293, 0.000000,0.270797,

iteration 3:
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.040125,-0.000000,-0.000000,0.000000,
0.244795,0.000000,0.108351,0.000000,0.100634,-0.002822,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.003904,0.000000,
0.000000,0.243139,0.000000,0.027883,0.000000,0.000000,0.000000,-0.000000,-
0.006543,0.000000,-0.000000,-0.000000,-0.000000,0.000000,
-0.018656,0.000000,-0.010398,0.000000,0.013373,0.192934,0.000000,0.000000,
0.000000,0.000000,-0.000000,-0.000000,-0.187737,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.242483,0.038131,
0.000000,0.022134,-0.000000,-0.056628,-0.000000,-0.108083,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.025548,0.059928,
0.000000,0.000001,0.000000,0.016800,-0.000000,-0.037522,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.385307,0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.042142,-0.026272,
0.000000,-0.016875,0.000000,0.481049, 0.000000,-0.004756,
-0.003138,-0.000000,-0.005941,-0.000000,-0.024330,-0.143058,-0.000000,-0.000000,-
0.000000,-0.000000,0.000000,0.000000, 0.465471,0.000000,
0.000000,0.000000,0.000000,-0.000000,0.000000,-0.000000,-0.102353,-0.015214,-
0.000000,-0.003714,-0.000000,-0.011543, 0.000000,0.265906,

iteration 4:
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.037712,-0.000000,-0.000000,0.000000,
0.236145,0.000000,0.104857,0.000000,0.097356,-0.003362,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.002732,0.000000,
0.000000,0.245101,0.000000,0.028861,0.000000,0.000000,0.000000,-0.000000,-
0.004839,0.000000,-0.000000,-0.000000,-0.000000,0.000000,

-0.018188,0.000000,-0.010109,0.000000,0.014225,0.197314,0.000000,0.000000,
0.000000,0.000000,-0.000000,-0.000000,-0.195943,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.248555,0.038279,
0.000000,0.021711,-0.000000,-0.058468,-0.000000,-0.111118,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.027106,0.061925,
0.000000,0.000148,0.000000,0.014406,-0.000000,-0.036890,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.373152,0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.043338,-0.026581,
0.000000,-0.017075,0.000000,0.478228, 0.000000,-0.001952,
-0.002502,-0.000000,-0.005562,-0.000000,-0.024790,-0.148408,-0.000000,-0.000000,-
0.000000,-0.000000,0.000000,0.000000, 0.471606,0.000000,
0.000000,0.000000,0.000000,-0.000000,0.000000,-0.000000,-0.107297,-0.015649,-
0.000000,-0.003703,-0.000000,-0.007560, 0.000000,0.264702,

iteration 5:
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.036163,-0.000000,-0.000000,0.000000,
0.231979,0.000000,0.103182,0.000000,0.095741,-0.003790,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.001913,0.000000,
0.000000,0.246332,0.000000,0.029311,0.000000,0.000000,0.000000,-0.000000,-
0.003887,0.000000,-0.000000,-0.000000,-0.000000,0.000000,
-0.018105,0.000000,-0.010141,0.000000,0.014514,0.199918,0.000000,0.000000,
0.000000,0.000000,-0.000000,-0.000000,-0.200379,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.252499,0.038493,
0.000000,0.021458,-0.000000,-0.059319,-0.000000,-0.112822,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.028204,0.063355,
0.000000,0.000222,0.000000,0.012897,-0.000000,-0.036533,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.365412,0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.043892,-0.026646,
0.000000,-0.017020,0.000000,0.475470, 0.000000,-0.000476,
-0.002001,-0.000000,-0.005192,-0.000000,-0.024875,-0.151727,-0.000000,-0.000000,-
0.000000,-0.000000,0.000000,0.000000, 0.475597,0.000000,
0.000000,0.000000,0.000000,-0.000000,0.000000,-0.000000,-0.110439,-0.016060,-
0.000000,-0.003707,-0.000000,-0.005307, 0.000000,0.263808,

iteration 6:
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.035214,-0.000000,-0.000000,0.000000,
0.229926,0.000000,0.102353,0.000000,0.094928,-0.004083,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.001449,0.000000,

0.000000,0.247199,0.000000,0.029562,0.000000,0.000000,0.000000,-0.000000,-
0.003304,0.000000,-0.000000,-0.000000,-0.000000,0.000000,
-0.018157,0.000000,-0.010248,0.000000,0.014611,0.201485,0.000000,0.000000,
0.000000,0.000000,-0.000000,-0.000000,-0.202742,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.255074,0.038669,
0.000000,0.021316,-0.000000,-0.059749,-0.000000,-0.114118,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.028905,0.064277,
0.000000,0.000259,0.000000,0.011981,-0.000000,-0.036368,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.360376,0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.044207,-0.026652,
0.000000,-0.016940,0.000000,0.473538, 0.000000,0.000432,
-0.001696,-0.000000,-0.004948,-0.000000,-0.024902,-0.153638,-0.000000,-0.000000,-
0.000000,-0.000000,0.000000,0.000000, 0.477681,0.000000,
0.000000,0.000000,0.000000,-0.000000,0.000000,-0.000000,-0.112635,-0.016415,-
0.000000,-0.003765,-0.000000,-0.003923, 0.000000,0.263687,

iteration 7:
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.034634,-0.000000,-0.000000,0.000000,
0.228993,0.000000,0.101968,0.000000,0.094530,-0.004301,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.001133,0.000000,
0.000000,0.247786,0.000000,0.029687,0.000000,0.000000,0.000000,-0.000000,-
0.002959,0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
-0.018265,0.000000,-0.010381,0.000000,0.014601,0.202450,0.000000,0.000000,
0.000000,0.000000,-0.000000,-0.000000,-0.204115,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.256734,0.038801,
0.000000,0.021235,-0.000000,-0.059939,-0.000000,-0.115030,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.029369,0.064913,
0.000000,0.000294,0.000000,0.011414,-0.000000,-0.036297,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.357111,0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.044360,-0.026625,
0.000000,-0.016863,0.000000,0.472056, 0.000000,0.000964,
-0.001472,-0.000000,-0.004760,-0.000000,-0.024868,-0.154802,-0.000000,-0.000000,-
0.000000,-0.000000,0.000000,0.000000, 0.478934,0.000000,
0.000000,0.000000,0.000000,-0.000000,0.000000,-0.000000,-0.114085,-0.016678,-
0.000000,-0.003825,-0.000000,-0.003108, 0.000000,0.263763,

iteration 8:
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.034276,-0.000000,-0.000000,0.000000,

0.228591,0.000000,0.101796,0.000000,0.094335,-0.004456,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.000924,0.000000,
0.000000,0.248196,0.000000,0.029754,0.000000,0.000000,0.000000,-0.000000,-
0.002744,0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
-0.018374,0.000000,-0.010499,0.000000,0.014563,0.203060,0.000000,0.000000,
0.000000,0.000000,-0.000000,-0.000000,-0.204913,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.257821,0.038899,
0.000000,0.021194,-0.000000,-0.060020,-0.000000,-0.115701,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.029671,0.065330,
0.000000,0.000321,0.000000,0.011068,-0.000000,-0.036273,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.354955,0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.044440,-0.026595,
0.000000,-0.016802,0.000000,0.470998, 0.000000,0.001294,
-0.001316,-0.000000,-0.004624,-0.000000,-0.024824,-0.155504,-0.000000,-0.000000,-
0.000000,-0.000000,0.000000,0.000000, 0.479654,0.000000,
0.000000,0.000000,0.000000,-0.000000,0.000000,-0.000000,-0.115070,-0.016873,-
0.000000,-0.003882,-0.000000,-0.002606, 0.000000,0.263947,

iteration 9:
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-0.000000,-
0.000000,-0.000000,-0.034052,-0.000000,-0.000000,0.000000,
0.228444,0.000000,0.101728,0.000000,0.094244,-0.004567,-0.000000,-0.000000,-
0.000000,-0.000000,-0.000000,-0.000000,-0.000781,0.000000,
0.000000,0.248483,0.000000,0.029788,0.000000,0.000000,0.000000,-0.000000,-
0.002610,0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
-0.018469,0.000000,-0.010595,0.000000,0.014518,0.203453,0.000000,0.000000,
0.000000,0.000000,-0.000000,-0.000000,-0.205399,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.258537,0.038969,
0.000000,0.021173,-0.000000,-0.060049,-0.000000,-0.116181,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.029870,0.065614,
0.000000,0.000343,0.000000,0.010850,-0.000000,-0.036269,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.353523, 0.000000, 0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.044478,-0.026567,
0.000000,-0.016756,0.000000,0.470233, 0.000000,0.001500,
-0.001205,-0.000000,-0.004526,-0.000000,-0.024780,-0.155940,-0.000000,-0.000000,-
0.000000,-0.000000,0.000000,0.000000, 0.480090,0.000000,
0.000000,0.000000,0.000000,-0.000000,0.000000,-0.000000,-0.115734,-0.017014,-
0.000000,-0.003929,-0.000000,-0.002296, 0.000000,0.264133,

Converging threshold Criteria:

Since it will be meaningless for the code to run for unlimited iterations, we have to fix the thresold criteria for the output. Here limitation is exercised on asdv value. If for nth iteration, the value stored in asdv, that gives average of sum of difference values between two successive iterations, is less than 0.0009, then the output will be of (n-1)th iteration and program will be terminated.

Final Output:

average sum of difference values of 1 iteraration: 0.006764
average sum of difference values of 2 iteraration: 0.002121
average sum of difference values of 3 iteraration: 0.001216
average sum of difference values of 4 iteraration: 0.000712
average sum of difference values of 5 iteraration: 0.000424
average sum of difference values of 6 iteraration: 0.000254
average sum of difference values of 7 iteraration: 0.000159
average sum of difference values of 8 iteraration: 0.000102
average sum of difference values of 9 iteraration: 0.000067
the similarity values are almost converged.
result after 9 iterations:
0.000000,0.000000,0.000000,0.000000,0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.034276,-0.000000,-0.000000,0.000000,
0.228591,0.000000,0.101796,0.000000,0.094335,-0.004456,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000924,0.000000,
0.000000,0.248196,0.000000,0.029754,0.000000,0.000000,0.000000,-0.000000,-0.002744,0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
-0.018374,0.000000,-0.010499,0.000000,0.014563,0.203060,0.000000,0.000000,
0.000000,0.000000,-0.000000,-0.000000,-0.204913,-0.000000,
-0.000000,0.000000,-0.000000,0.000000,0.000000,0.000000,0.257821,0.038899,
0.000000,0.021194,-0.000000,-0.060020,-0.000000,-0.115701,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.029671,0.065330,
0.000000,0.000321,0.000000,0.011068,-0.000000,-0.036273,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,
0.000000,-0.000000,0.354955,0.000000,0.000000,-0.000000,
-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.000000,-0.044440,-0.026595,
0.000000,-0.016802,0.000000,0.470998,0.000000,0.001294,
-0.001316,-0.000000,-0.004624,-0.000000,-0.024824,-0.155504,-0.000000,-0.000000,-0.000000,-0.000000,0.000000,0.000000,0.479654,0.000000,
0.000000,0.000000,0.000000,-0.000000,0.000000,-0.000000,-0.115070,-0.016873,-0.000000,-0.003882,-0.000000,-0.002606,0.000000,0.263947,

Operating System

Introduction to Unix:

UNIX was originally developed at Bell Laboratories as a private research project by a small group of people starting in 1969. This group had experience with a number of different operating systems research efforts in the 1970's. The goals of the group were to design an operating system to satisfy the following objectives:

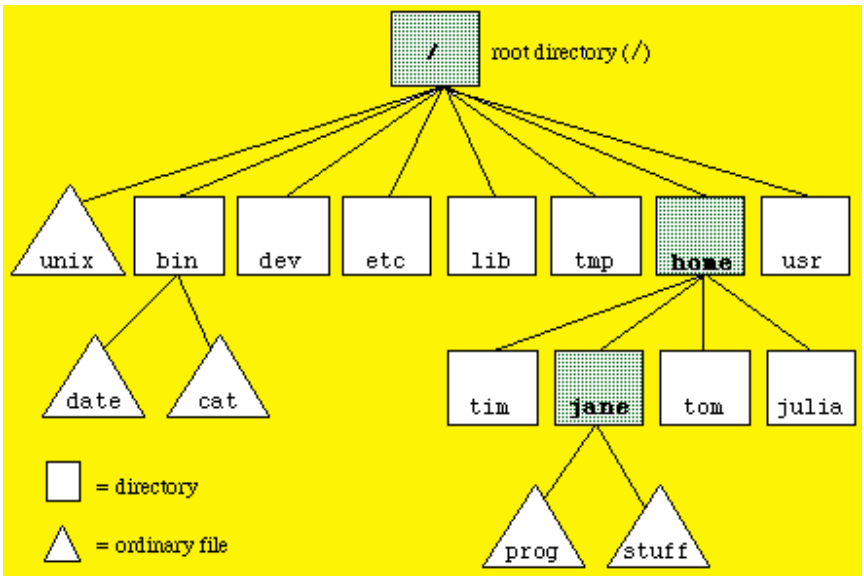
- Programmers environment
- Simple user interface
- Simple utilities that can be combined to perform powerful functions
- Hierarchical file system
- Simple interface to devices consistent with file format
- Multi-user, multi-process system
- Architecture independent and transparent to the user.

The UNIX system is mainly composed of three different parts: the kernel, the file system, and the shell.

> [The kernel] is that part of the system which manages the resources of whatever computer system it lives on, to keep track of the disks, tapes, printers, terminals, communication lines and any other devices.
>
> The file system is the organising structure for data. The file system is perhaps the most important part of the UNIX operating system. The file system goes beyond being a simple repository for data, and provides the means of organizing the layout of the data storage in complex ways.

Diagram 9

File system of UNIX



The shell is the command interpreter. Although the shell is just a utility program, and is not properly a part of the system, it is the part that the user sees. The shell listens to your terminal and translates your requests into actions on the part of the kernel and the many utility programs.

One can imagine the UNIX system as a series of three concentric circles, with the inner circle representing the kernel, the second circle representing the programming shell, and the last one representing application programs.

Multi-tasking, Time Sharing

UNIX is a multi-tasking operating system, which means that a number of programs can run at the same time. Those programs (called processes) can communicate with each other. For example, a C program could be compiling as mail is being read or a file is

being edited. Processes that ``wake-up'' occasionally, and/or regularly, are called daemons. Daemons are used to synchronize disks, send and receive mail, print documents and so on.

Multi-user

UNIX is also multi-user: two, three, or more users are able to use the same processor to execute their programs.

Network Capabilities

Today's UNIX workstations come with TCP/IP and ethernet connections. The same is true for PCs. At NRC the ethernet network connects dozens of Suns, Silicon Graphics, VAXes, IBM RS/6000s, HPs, and most PCs together. From any of those nodes, it is simple to logon to a remote machine, send mail to a user on those machines, or transfer files to or from those nodes.

Portability

Traditionally, most operating systems were written in Assembler, for a specific architecture. It was therefore VERY painful - if at all possible - to ‘port’ the operating system to other architectures. UNIX, on the other hand, is mostly written in the C language. This alone allows UNIX to be portable to many architectures. Today, UNIX/linux runs on more architectures than any other operating system in the world. Examples of such architectures/processors are the Motorola 680X0-based workstations, the 80X86 machines, the RISC based architectures (SPARC, MIPS, 88000), VAXes, IBM mainframes, Amdahl, Cray, and many more. And this does not include all the different architectures linux runs on.

Flexibility

UNIX is also a very flexible operating system, both for system administrators and users. Program names can be changed, aliases can be defined. Arguments to programs can also

be changed. New programs can be built, and put in the user's own `bin` directory, thus allowing further customization of the system

Virtual Memory

The UNIX operating system has virtual memory, or swap space, which means it can run programs bigger than the amount of RAM the computer actually has! The amount of virtual memory is decided upon by the system administrator.

Case Sensitivity

The most common mistake for beginners involves the use of mixed case in UNIX commands: UNIX is case sensitive, i.e., ``a" is different from ``A".

Software Available

Thousands of application packages are available for the UNIX/Linux system. In addition to the commercial packages, many programs are written and made available in the public domain. Examples of such packages are the X Window system, written at the Massachusetts Institute of Technology, the TeX system produced at Stanford, and many other utilities/applications written by individuals and organizations, for the benefit of the ``Open Source Community". For many, it is their way of thanking the ``Internet Community" for the vast amount of resources available.

Linux is a perfect example of the incredible amount of software available, at no cost to individuals. In fact, most open source, freeware, or public domain packages used in the scientific world are developed and maintained on a Linux platform. The same can be said of any open source, freeware, or public domain packages developed on any UNIX variant: Linux is the platform of choice to develop/write software.

UNIX Philosophy

UNIX's philosophy is the same as that of the C language; it assumes users know what they are doing.

LANGUAGE SPECIFICATION:

C was developed by Dennis Ritchie at AT & T's Bell Laboratories of USA in 1972. It's a mixture of BCPL (Basic Combined Programming Language) developed by Martin Richards at Cambridge University, B written by Ken Thompson at AT & T's Bell laboratories and its inventor Ritchie's own ideas.

A programming language can be termed as Problem oriented language or high level language like FORTRAN, BASIC, PASCAL etc. or Machine oriented language or low level language like Assembly languages and Machine languages. C is in between these two categories as it contains both; a relatively good programming efficiency and relatively good machine efficiency.

Common C

Until recently there was one dominant form of the C language. This was the native UNIX form, which for historical reasons is known as either Bell Labs C, after the most popular compiler, or K. &R. C, after the authors of the most popular textbook on the language. It is now often called "Classic C".

ANSI C

The American National Standards Institute defined a standard for C, eliminating much uncertainty about the exact syntax of the language. This newcomer, called ANSI C, proclaims itself the standard version of the language. As such it will inevitably overtake, and eventually replace common C. ANSI C does incorporate a few improvements over the old common C. The main difference is in the grammar of the language. The form of

function declarations has been changed making them rather more like Pascal procedures. Most C programming texts are now available in ANSI editions.

Using C with UNIX

A little knowledge is necessary before one can write and compile programs on the UNIX system. Every programmer goes through the same three-step cycle.

1. Writing the program into a file
2. Compiling the program
3. Running the program

During program development, the programmer may repeat this cycle many times, refining, testing and debugging a program until a satisfactory result is achieved.

Writing the Program

UNIX expects the user to store the program in a file whose name ends with ".c" extension. This identifies it as a C program. The easiest way to enter text is using a text editor like *vi*, *emacs* or *xedit*. To edit a file called exp.c using vi type

vi exp.c

The editor is also used to make subsequent changes to the program.

Compiling the Program

There are a number of ways to achieve this, though all of them eventually rely on the compiler (called cc on our system).

The C Compiler (cc)

The simplest method is to type

cc exp.c

This will try to compile exp.c and if successful, will produce a runnable file called a.out. If the user wants to give the runnable file a better name he/she can type

cc exp.c -o exp

This will compile exp.c, creating runnable file exp.

Make a Program Builder

UNIX also includes a very useful program called make. Make allows very complicated programs to be compiled quickly, by reference to a configuration file (usually called Makefile). If the C program is a single file, one can usually use make by simply typing

make exp

This will compile exp.c and put the executable code in exp.

Improved Type Checking Using Lint

The C compiler is rather liberal about type checking function arguments it doesn't check bounds of array indices. There is a stricter checker called lint, which won't generate any runnable code. It is a good idea to use lint to check the programs before they are completed. This is done by typing

lint exp.c

Lint is very good at detecting errors, which cause programs to crash at run time. However, lint is very fussy, and generally produces a long list of messages about minor problems with the program. Many of these will be quite harmless. Experience will only teach to distinguish the important messages from those, which can be ignored.

Running the Program

To run a program under UNIX the user simply type in the filename. So to run program exp, one would type

exp

or if this fails to work, one could type

./exp

Prompt will again be seen after the program is done.

UNIX File Redirection

UNIX has a facility called redirection, which allows a program to access a single input file and a single output file very easily. The program is written to read from the keyboard and write to the terminal screen as normal.

To run prog1 but read data from file infile instead of the keyboard, one would type

prog1 < infile

To run prog1 and write data to outfile instead of the screen, one would type

prog1 > outfile

Both can also be combined as in

prog1 < infile > outfile

Redirection is simple, and allows a single program to read or write data to or from files or the screen and keyboard.

Some programs need to access several files for input or output, redirection cannot do this. In such cases we have to use C's file handling facilities.

UNIX Library Functions

The UNIX system provides a large number of C functions as libraries. Some of these implement frequently used operations, while others are very specialized in their application.

Finding Information about Library Functions

The UNIX manual has an entry for all available functions. Function documentation is stored in section 3 of the manual, and there are many other useful system calls in section 2. If one already knows the name of the function he/she wants, can read the page by typing

man 3 functionname

If the user doesn't know the name of the function, a full list is included in the introductory page for section 3 of the manual. To read this, one has to type

man 3 intro

There are approximately 700 functions described here. This number tends to increase with each upgrade of the system.

On any manual page, the SYNOPSIS section will include information on the use of the function. The DESCRIPTION section will then give a short description of what the function does. Further related reading is suggested in the SEE ALSO section.

Use of Library Functions

To use a function, one has to ensure that he/she has included the required header file in the C file. Then the function can be called. It is important to ensure that the arguments have the expected types otherwise the function will probably produce strange results. lint is quite good at checking such things.

Some libraries require extra options before the compiler can support their use. For example, to compile a program including functions from the math.h library the command might be

cc exp.c -o exp -lm

The final -lm is an instruction to link the maths library with the program. The manual page for each function will usually inform the user if any special compiler flags are required.

Daigram 10

Some Useful Library Functions

Function	Description
abs	integer absolute value
ctime	convert date and time
fopen	open a stream
printf	formatted output
fputc	put character or word on a stream
getwd	get current working directory path name
strcat	string concatenation
ctype	character classification and conversion macros and functions
mktemp	make a unique file name
puts	put a string on a stream
sleep	suspend execution for interval
stdio	standard buffered input/output package

SOFTWARE SPECIFICATION:

Exceed 6.0 (The common desktop environment)

Exceed is a tool, to display remote X Window System applications on Microsoft Windows. It permits applications, normally available only on expensive UNIX workstations, to be readily accessed by an existing Windows NT, Windows 95/98, and Windows 3.x-based personal computers.

Exceed 6.0 is a 32 bit, fully X11R6-compliant X server and collection of programs and utilities that extend the capabilities of PC and let it communicate with Unix systems running X-Windows. The current version of Exceed (version 6.0 for Windows NT and Windows 95 or version 5.2 for DOS and Windows 3.11) supports NT on Appha systems

and all versions of Windows on intel. To run the software, one need at least 16 mB of RAM and a minimum of 90MB of available hard disk space.

Exceed operates over a variety of network protocol links, but generally Winsock-compliant network connection is used to establish link and execute programs. For example, for DOS and Windows 3.11, one can use Microsoft's TCP/IP-32 version 3.11b or another IP stack, such as FTP software's PC/TCP. On NT and Win95 systems, simply TCP/IP protocol, that the operating system includes, is used. Getting Exceed up and running in test environment takes less than 20 minutes. The distribution software comes on CD-ROM. After installation the system have to be rebooted.

While launching the installation program, the software runs an installation wizard to step through the process. The important item to know is which fonts are to be installed for use with the software. The product includes many categories of fonts; 75dpi is the default. Depending on the X client one might to need install additional font sets. One must also select the keyboard layout intended for use.

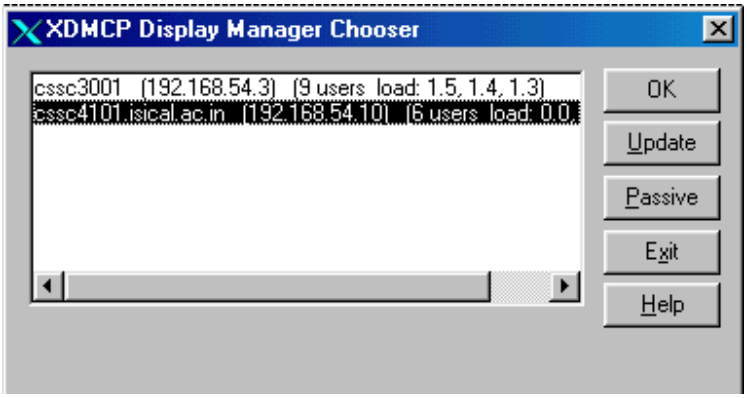
After all files are copied, the installation wizard asks whether the performance tuner be run or not. Running this program lets Exceed test the computer's video configuration to find the best way to render graphics and perform other graphical operations. This process may be lengthy, depending on speed of computer's CPU and video subsystem.

Accessing the X Window system using Exceed

From a Windows workstation, Exceed can be run by double clicking the shortcut to Exceed icon or selecting it from the Start menu. From XDMCP display manager chooser window the user have to select a group appropriate for him/her and press OK. For example the user here is a member of the cssc4101.isical.ac.in (192.168.54.10) group.

Diagram 11

XDMCP Display Manager Chooser



Next a window will appear displaying Welcome to cssc4101, urging the user to enter user-id and password and press OK. If any of these two got entered wrong, the user has to enter both user-id and password again.

Diagram 12

Password Window

Once logged in successfully, the user either have to transfer already written program to his account from hard disk memory of the workstation or can write the program using

text editor. But the code should be X-Window suitable as it'll be compiled and run in the X-Window only.

Diagram 13

Anonymous FTP

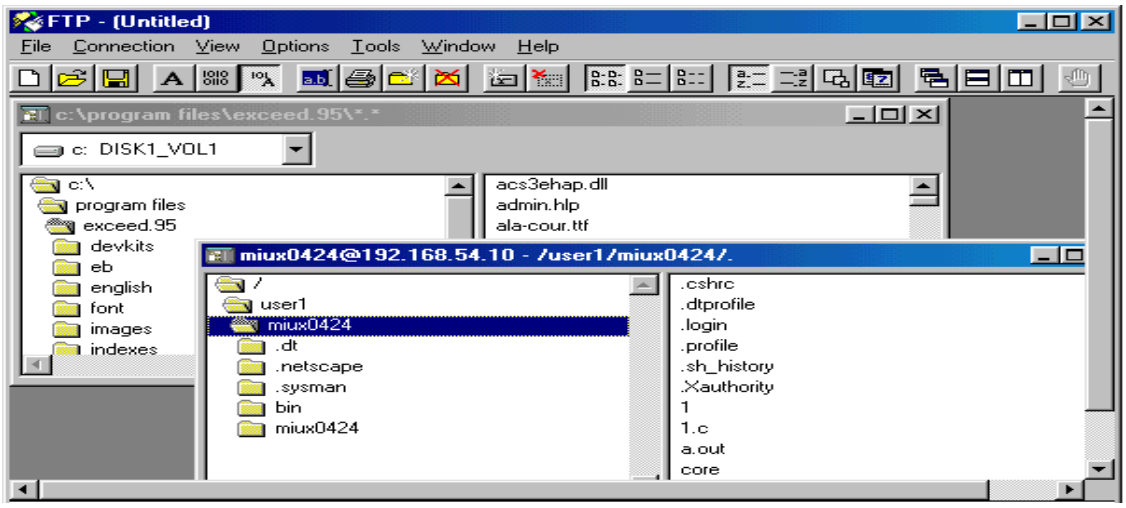
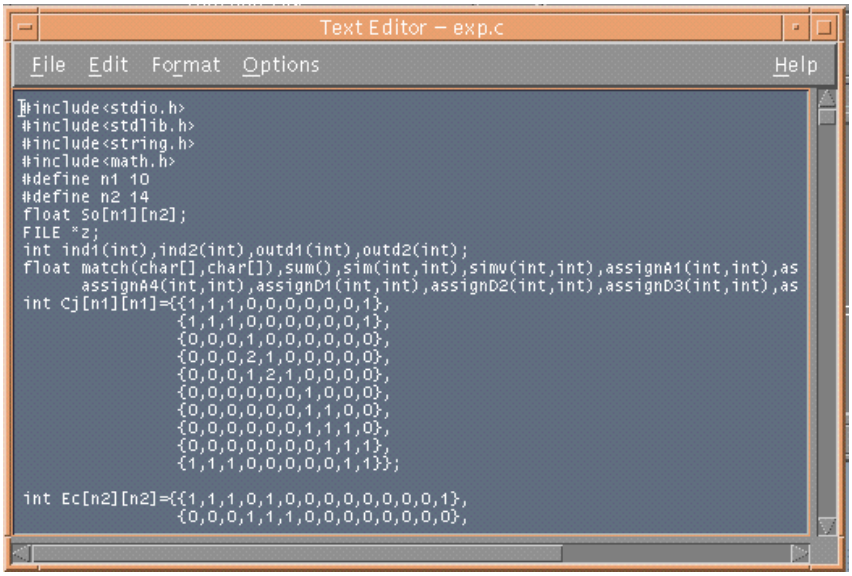


Diagram 14

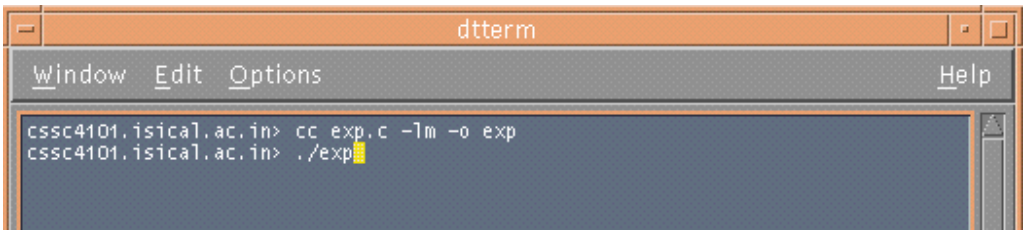
Text Editor



The program is compiled and run in the terminal window. One can open more than one widows simultaneously, be it text editors or terminals. But it is advisable to open the minimum no. of windows at a time.

Diagram 15

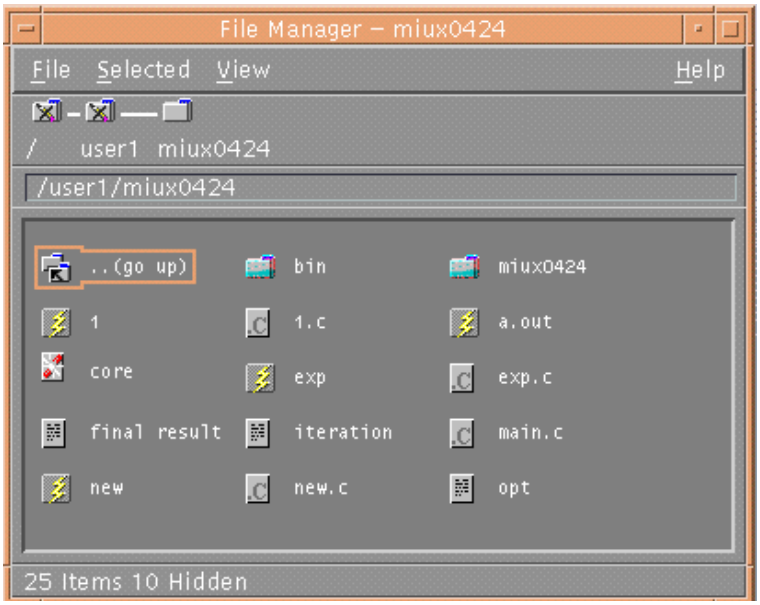
The Terminal



Folders and files created can be edited and arranged through file manager.

Diagram 16

File Manager



After use the session can be ended by logging out or simply closing the exceed window.

CONCLUSION:

Here I have tried to compare two species-specific metabolic maps of TCA Cycle. For comparison, adjacency matrix corresponding to the metabolic maps, are used. Among the four rules involved in creation of adjacency matrices, two are newly implemented, one is modified and the rest one is taken from the original paper (Heymans and Singh, 2003). Also the graphs created by Heymans and Singh are not up to dated according to the updation of KEGG database. So the enzyme graphs obtained are not expected to bear total resemblance with enzyme graphs created by the authors.

After implementation of the first step of the algorithm in C code, we get a matrix with float values ranging from –1 to 1 that explains level of similarity between a pair of nodes, given the condition each one of them belong to different enzyme graph.

This resulting matrix values can be used further for Bipertite graph matching, calculation of final similarity score; conversion of similarity scores to distance scores as well as phylogram construction according to the algorithm. Complete implementation of the algorithm and its large-scale application for phylogenetic analysis of distantly related species can throw light on trends of evolution from metabolic point of view.

Bibliography:

1. http://www.fact-index.com/m/me/metabolic_pathway.html
2. http://www.swbic.org/links/1.5.php
3. http://www2.ufp.pt/~pedros/bq/integration.htm
4. http://ull.chemistry.uakron.edu/Pathways/
5. http://vlib.org/Science/Cell_Biology/metabolism.shtml#databases
6. http://www-mbi3.kuicr.kyoto-u.ac.jp/~kihara/biolink/pathdb.html

7. http://www.genome.jp/kegg/kegg4.html
8. http://www.all-science-fair projects.com/science_fair_projects_encyclopedia/EC_number
9. http://www.chem.qmw.ac.uk/iubmb/enzyme/
10. http://www.nature.com/nature/journal/v420/n6915/fig_tab/nature01266_F3.html
11. http://www.strath.ac.uk/IT/Docs/Ccourse/
12. http://www.winnetmag.com/Windows/Article/ArticleID/2996/2996.html
13. http://computing-dictionary.thefreedictionary.com/Exceed
14. http://www.genome.ad.jp/kegg/
15. http://encyclopedia.thefreedictionary.com/Encyclopedia
16. http://www.chem.qmw.ac.uk/iubmb/enzyme/
17. K.,Minoru, G.,Susumu, " KEGG: Kyoto encyclopedia of Genes and Genomes", Institute of Chemical Research, Kyoto University, Japan, *Nucleic Acid Research, 2000, Vol. 28, No. 1*.
18. F., A., David, W. Andreas, "*METABOLIC* ENGINEERING: The small world of metabolism", © 2000 Nature America Inc. http://biotech.nature.com
19. O., Hiroyuki, B., Hidemasa, F., Wataru, G., Susumu, K., Minoru, "Analysis of Binary Relations and Hierarchies of Enzymes in the Metabolic Pathways", Institute of Chemical Research, Kyoto University, Japan.
20. K., Minoru, G., Susumu, K., Shuichi, O., Yasushi, H., Masahiro, "The KEGG resource for deciphering the genome", Bioinformatics Center, Institute for Chemical Research, Kyoto University, Japan.
21. Maureen, S., K., Aambuj, "Deriving phylogenetic trees from the similarity analysis of metabolic pathways", Department of Computer Science, University of California, Santa Barbara, USA, *Bioinformatics, 2003, vol. 19, suppl. 2, pages i138-i146.*