



Title	Development of computational method for heterologous pathways design
Author(s)	Chatsurachai, Sunisa
Citation	大阪大学, 2013, 博士論文
Version Type	VoR
URL	https://doi.org/10.18910/26195
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

Doctoral Dissertation

Development of computational method for heterologous pathways design

Sunisa Chatsurachai

July 2013

**Department of Biotechnology
Graduate School of Engineering,
Osaka University**

Table of Contents

Chapter 1	1
General introduction	1
1.1 Importance of bio-based process for valuable chemicals	2
1.2 Importance of computational methods for metabolic pathway design	7
1.3 Modeling and analysis of genome-scale metabolic networks	17
1.4 Microorganisms used as industrial cell factories	27
1.5 Objectives of the work	30
1.6 Outline of the thesis	32
Chapter 2	35
Development of an algorithm to design heterologous pathway	35
2.1 Introduction	35
2.2 Methods	37
2.2.1 Construction of an in-house database of metabolic reactions	37
2.2.2 Genome-scale metabolic model of host microorganisms	37
2.2.3 Heterologous pathway identification for target production	38
2.2.4 Flux balance analysis (FBA)	40
2.3 Results and discussion	41
2.3.1 Identification of heterologous pathway(s)	41

2.3.2 Evaluation of production feasibility	45
2.3.3 Differences in target production capacity among host microorganisms	53
2.4 Summary	56
Chapter 3	58
Selection of heterologous genes using CAI score.....	58
3.1 Introduction	58
3.2 Materials and methods	60
3.2.1 Constructing an in-house database of metabolic reactions	60
3.2.2 Screening for artificial heterologous pathways involved in the	61
production of targets	
3.2.3 Codon Adaptation Index (CAI)	61
3.3 Results and discussion	64
3.3.1 Identification of heterologous pathways	64
3.3.2 Relationship between CAI score and protein abundance	66
3.3.3 Screening of heterologous genes with higher CAI scores	67
3.3.4 Examples of results in hyper-text based user interface	73
3.3.5 Examples of new heterologous pathways for the production of.....	79
nonnative metabolites in the specific host	
3.4 Summary	84

Chapter 4	86
General conclusion and Future perspective.....	86
4.1 General conclusion and discussion	86
4.2 Future perspective	92
References	98
Appendix A source code	110
Appendix B	142
List of publications	143
Acknowledgements.....	144

Chapter 1

General introduction

Biosynthesis of biofuels, diverse chemicals and sustainable synthesis of several chemicals has attracted much attention due to the potential depletion of petroleum. Natural organisms often produce target metabolites at low yields, and it is difficult to improve bio-products since the information about metabolic system and genetic manipulation tools of the organisms are generally limited.

Metabolic engineering is the key solution broadly used to redirect existing metabolic pathways and/or to incorporate heterologous pathways into well-characterized hosts, including *Escherichia coli*, *Saccharomyces cerevisiae*, and so forth, for improvement of the productions of native and/or nonnative metabolites. However, it is still difficult to know and answer the following questions; How does the global metabolism of a microbial cell respond to changes in its environment? How do we get the feasible heterologous pathways and genes to improve the productivity from huge number of possible pathways/genes? Which host is suitable for a target production? To answer these questions, an appropriate computational method/*in silico* platform focusing on metabolic design is desirable.

This chapter, the importance of bio-based process and the reason for the development of *in silico* platform will be discussed. Then, several pathway design methods previously reported will be explained and key features comparing with this

thesis are summarized. In this study, flux balance analysis (FBA) was used to estimate fluxes of the metabolic system. In the third part of this chapter, a process for constructing a model of the metabolic system and how to use FBA technique for analyzing the metabolic fluxes will be presented. Finally, the objective and outline of this thesis will be summarized.

1.1 Importance of bio-based process for valuable chemicals

The demand of crude oil is gradually increasing as shown in Fig. 1.1 as well as increasing in its prices. These data suggest that petroleum sources are unsustainable. It takes about hundred to million years to make oil by decomposition of dead organisms mostly zooplankton and algae. Fuels and chemicals are mainly produced by petroleum-based process from crude oil. Moreover, the petroleum-based process has a negative impact on earth by releasing pollutants and generating hazardous wastes. Thus alternative energy resources, such as solar, wind, hydroelectric and biomass, which are renewable and sustainable, have been increasingly used in order to preserve fossil resources and to reduce CO₂-emissions. Among renewable energies, biomass shares the most consumption in 2010 (Fig. 1.2). Biomass can be converted into three main types of products, i.e., electrical/heat energy, transport fuels, and chemical feedstocks (McKendry, 2002) by using microorganism via bio-based processes.

In addition, biomass is available as renewable resources either as natural processes (wood, pulp) or as by-products and/or wastes of human activities

(molasses, rice straw, corn stover). Therefore, bio-based process by using microorganisms as cell factories to convert biomass into valuable products is becoming an attractive way. In many years ago, bio-based process mainly focused on improving the production of native metabolites (found in a host cell) widely used in food and beverage industry, for example, amino acids produced by *E. coli* and ethanol produced by yeast. Nowadays, trend of bio-based process for useful products has moved to produce nonnative metabolites (not found in the host cell) basically important as chemicals used for many industrial purposes such as polymers, pharmaceuticals, fuels, solvents, and so forth. Currently, those of useful native and nonnative metabolites are able to be produced by microorganisms (Dugar and Stephanopoulos, 2011; J. W. Lee et al., 2011; S. K. Lee et al., 2008; Papini et al., 2010; Schneider and Wendisch, 2011). The examples of such compounds and global demand are shown in Table 1.1. Some of bio-fermentation products, for instance, ethanol and higher alcohols are usually being used as fuels and solvents (Wang et al., 2012). 1,3-propanediol forms the basis of polymers such as polytrimethylene terephthalate (PTT) (H. Liu et al., 2010), while isoprene is an intermediate metabolite in the production of cis-1,4-polyisoprene, a synthetic version of natural rubber (Ohya and Koyama, 2005).

As shown in Fig. 1.3, the sales of chemicals made by bio-based process in 2010 were \$118 billion. It is forecasted to increase to \$296 and \$668 billion by 2015 and 2020, respectively. These data suggest the importance of industrial chemicals made by using bio-based process. The bio-based process for chemicals would be an efficient route for reducing wastes and CO₂ emissions, and preserving fossil fuels comparing with petroleum-based process.

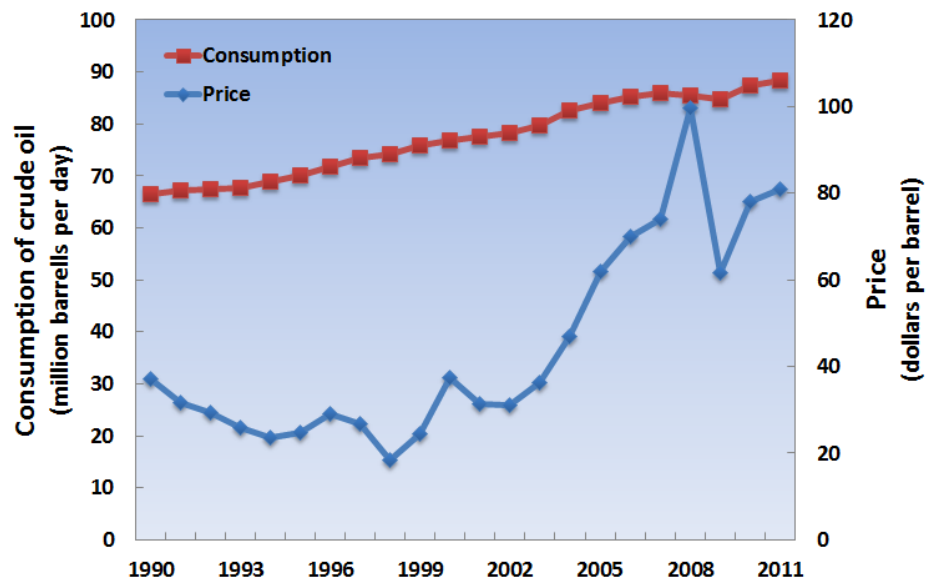


FIG. 1.1 Crude oil consumption and price from 1990-2011

(Source: U.S. Energy Information Administration available at <http://www.eia.gov/>)

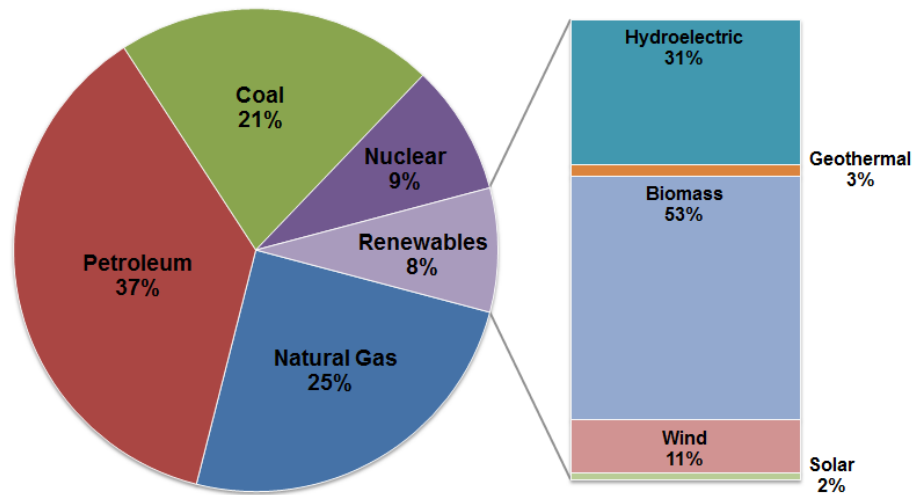


FIG. 1.2 World total and energy consumption by source, 2010

(Source: U.S. Energy Information Administration available at <http://www.eia.gov/>)

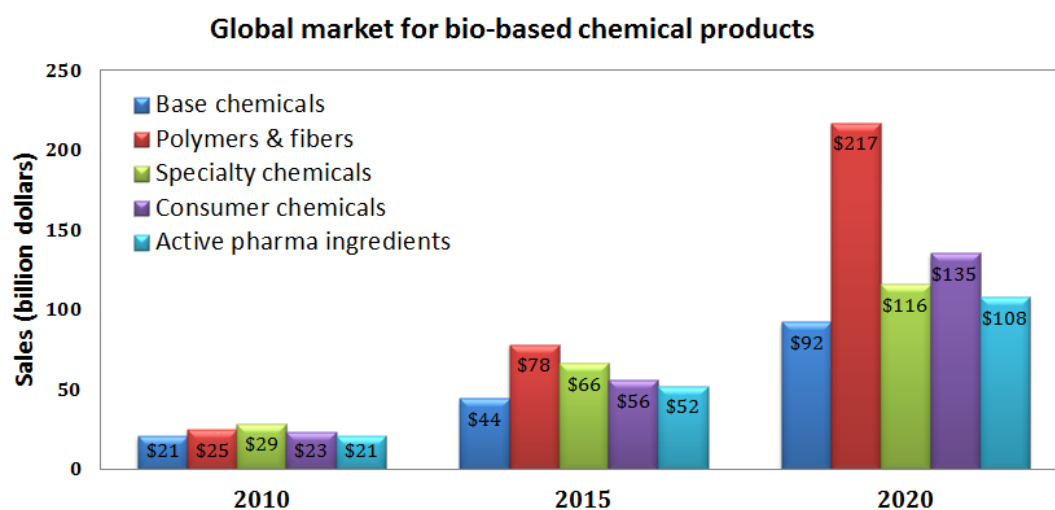


FIG. 1.3 Trends in production of bio-based chemicals. 2015 and 2020 data are projections based on 2010 data (Adapted from (Festel et al., 2012))

Table 1.1 Bio-based products and global demand

Compound	Host cell factory	Reference	Global market size (million ton/year)	Industrial application
1,3-propanediol	<i>Escherichia coli</i>	(H. Liu et al., 2010)	0.1-0.5	Co-polymers to produce PTT for plastics
	<i>Saccharomyces cerevisiae</i>	(Rao et al., 2008)		
1,4-butanediol	<i>Escherichia coli</i>	(Yim et al., 2011)	1.7	Polymers, solvents
1-butanol	<i>Escherichia coli</i>	(Shen et al., 2011)	3.0	Polymers, plastics, solvents
	<i>Synechococcus elongatus</i> PCC7942	(Lan and Liao, 2012)		
2,3-butanediol	<i>Escherichia coli</i>	(S. Lee et al., 2012)	0.06	Chemical, food, fuel fibers, plastics
	<i>Bacillus subtilis</i>	(Biswas et al., 2012)		
3-hydroxypropanoic acid	<i>Escherichia coli</i>	(Rathnasingh et al., 2012)	0.5	Contact lenses, polymers for diapers, carpet fibers
Cadaverine	<i>Corynebacterium glutamicum</i>	(Kind and Wittmann, 2011)	0.1	Polyamides for plastics
	<i>Escherichia coli</i>	(Z.-G. Qian et al., 2011)		
Ethanol	<i>Saccharomyces cerevisiae</i>	(Guadalupe Medina et al., 2010)	60	Biofuel, food beverages, solvents
	<i>Escherichia coli</i>	(Woodruff et al., 2013)		
Glucaric acid	<i>Escherichia coli</i>	(Moon et al., 2009)	0.06	Solvents, nylons
Glutamic acid	<i>Corynebacterium glutamicum</i>	(Becker and Wittmann, 2012)	2.5	Monomers for polyester and polyamides
Isoprene	<i>Escherichia coli</i>	(Lv et al., 2012)	0.1-0.5	Natural rubber, thermoplastics
Itaconic acid	<i>Aspergillus terreus</i>	(Kuenz et al., 2012)	0.08	Polymers, fibers
Lactic acid	<i>Escherichia coli</i>	(Mazumdar et al., 2013)	0.3-0.5	Polymers, plastics, fibers
	<i>Saccharomyces cerevisiae</i>	(Pacheco et al., 2012)		
	<i>Synnechocystis</i> sp. PCC6803	(Angermayr et al., 2012)		
Malic acid	<i>Escherichia coli</i>	(Zhang et al., 2011)	0.06	Acidulent in food industry
Muconic acid	<i>Saccharomyces cerevisiae</i>	(Curran et al., 2013)	2.30	Polymers, plastics
	<i>Escherichia coli</i>	(Niu et al., 2002)		
Succinic acid	<i>Corynebacterium glutamicum</i>	(Litsanov et al., 2012)	0.1	Feed additives, fuel additives, fibers, polymers
	<i>Escherichia coli</i>	(Hoefel et al., 2012)		
	<i>Saccharomyces cerevisiae</i>	(Otero et al., 2013)		

*Data retrieved from (Vennestrøm et al., 2011) and IEA Bioenergy (source:

www.ieabioenergy.com/) .

1.2 Importance of computational methods for metabolic pathway design

Several approaches leading to the production of valuable products generally use engineered microbes in which native metabolic networks of microorganisms are artificially modified to produce target products. One such standard strategy employed for producing target metabolites is the incorporation of heterologous pathways into well-characterized hosts such as *B. subtilis*, *C. glutamicum*, *E. coli*, and *S. cerevisiae* as shown in Table 1.1. Nevertheless, the selection of suitable heterologous metabolic pathways for host organisms is often difficult due to the metabolic network complexity. Besides, huge amount of information on metabolic reactions have been found in literature, and are available on public databases such as KEGG (Kanehisa et al., 2008), BRENDA (Chang et al., 2009), ENZYME (Bairoch, 2000), MetaCyc (Caspi et al., 2008), and BKM-react (Lang et al., 2011) as shown in Fig. 1.4. There are 8,507, 8,244, and 9,096 metabolic reactions available on BRENDA, KEGG, and MetaCyc, respectively, and the integration database, BKM-react, contains 18,172 unique metabolic reactions combining from the 3 databases, BRENDA, KEGG and MetaCyc. Thus, to find appropriate heterologous pathways for target production generally requires massive calculations. For example, to search for heterologous pathway for producing target nonnative metabolite, it requires users to search the heterologous reactions which are able to connect nonnative metabolite to the target host metabolism. As you can see in Fig. 1.5, there are 11 possible routes/heterologous pathways containing 4 heterologous reactions to connect nonnative metabolite to the host's metabolic network. Since the complexity of

metabolic network available on database, it is difficult to manually check and search for all possible heterologous pathways. In addition, the substrate-product conversion should be primitively estimated as a reference value to be compared with experimental outcome, and could be used to evaluate the feasibility of heterologous pathways.

In Fig. 1.6, a toy metabolic network demonstrates 7 biochemical reactions (2 intracellular and 5 transport reactions) and contains 5 intracellular metabolites (A-E) and 5 extracellular metabolites (A_{ext} , C_{ext} , D_{ext} , E_{ext} , and Biomass). If biomass is a target from this network, theoretical yield can be calculated from reaction stoichiometry between reactants and products. Therefore, 1 mole of A, D, and C will be converted to 2 mole of Biomass and 1 mole of E as a by-product as shown in Fig. 1.7. This toy metabolic network is simple and contains only 7 reactions, so it is possible to calculate by hand. However, the real host cell such as *E. coli* contains 931 internal and 143 transport reactions (Reed et al., 2003), it may be impossible to manually calculate theoretical yield of target product and find feasible pathways of targets.

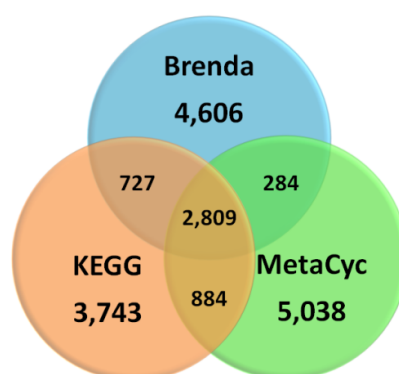


FIG. 1.4 Distribution of the unique metabolic reactions between BRENDA, KEGG, and MetaCyc databases (Lang et al., 2011)

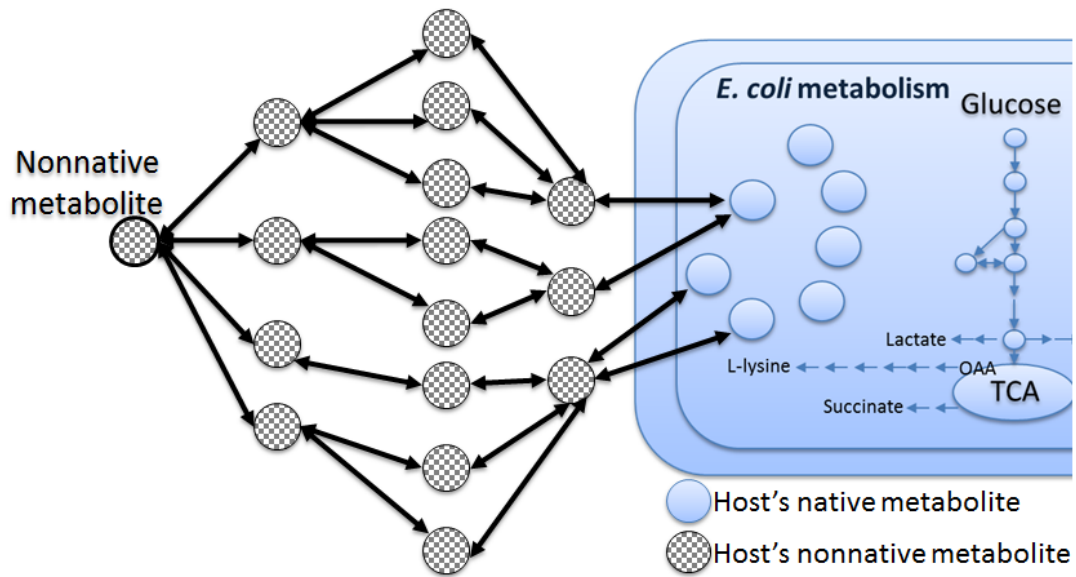


FIG. 1.5 Heterologous pathway design by human to search for possible heterogeneous reactions to connect nonnative metabolite to the host metabolism

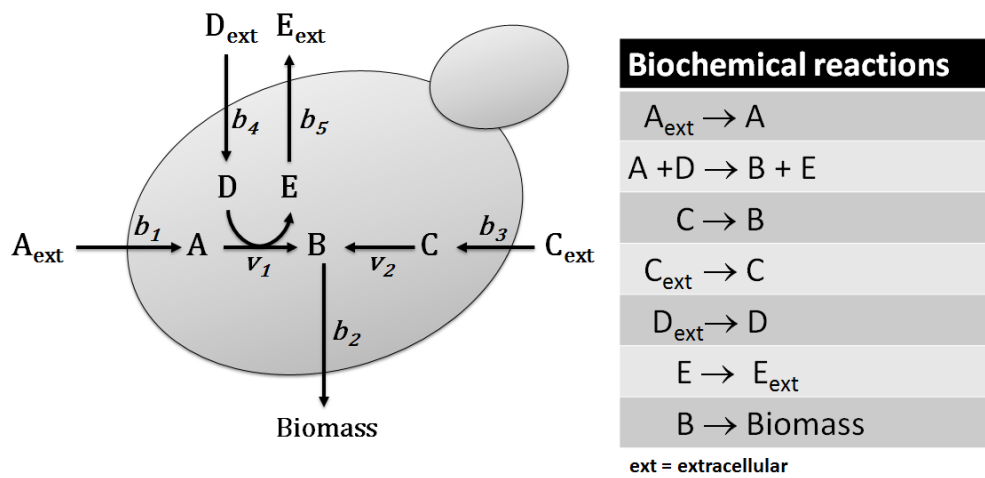


FIG. 1.6 Toy metabolic network map consists of 5 intracellular metabolites (A-E) and 5 extracellular metabolites (A_{ext} , C_{ext} , D_{ext} , E_{ext} , and Biomass), 2 intracellular reactions (v_1 - v_2) and 5 transport reactions (b_1 - b_5) and biochemical reactions present on right-hand side.

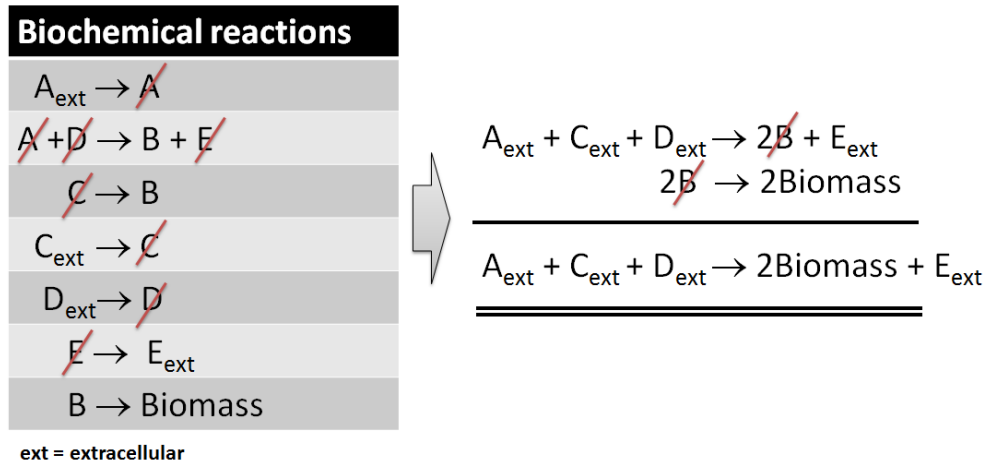


FIG. 1.7 Theoretical yield calculation based on reaction stoichiometry

Without using the computational method, it is difficult and tight to handle numerous reactions in host's metabolic network and to search heterologous reactions for the production of nonnative metabolites from databases. Additionally, to find the feasible heterologous pathway and to check whether one target nonnative metabolites produced by the host cell, are time-consuming tasks, when these tasks depend only on the researcher's knowledge and manual screen of information. From these reasons, a computational method is required to help researchers for finding the feasible heterologous pathway of target nonnative metabolite.

Recently, numerous *in silico* heterologous pathway search methods have been proposed and used in target metabolites productions (Cho et al., 2010; Dogrusoz et al., 2009; Finley et al., 2009; Flórez et al., 2011; Handorf et al., 2005; Li et al., 2004; McShan et al., 2003; Moriya et al., 2010; Pey et al., 2011; Pharkya et al., 2004; Rodrigo et al., 2008; Yousofshahi et al., 2011). Comparison of key points among those methods is shown in Table 1.2. The key points are composed of:

1. There is no requirement of background information for searching pathway, for example, rules of enzyme transformation between substrate to product.
2. The calculation time of the method is fast when search for the optimized pathway with maximum theoretical yield of target.
3. New/alternative pathways of target can be generated.
4. New reaction(s) not existing in the available databases can be generated.
5. The information of new reaction such as gene, sequence, protein is available.
6. The heterologous pathway is generated for specific host cells.
7. The information of heterologous genes corresponding to a heterologous pathway is available and specific for the target host cell.
8. All possible nonnative metabolites and heterologous pathways information are available as a catalog for each target host cell.

In Table 1.2, all approaches are able to screen the new pathway for target production. However, only PathMiner (McShan et al., 2003), Pathway generation (C. Li et al., 2004), PathPred (Moriya et al., 2010), BNICE (Finley et al., 2009), and Prioritization (Cho et al., 2010). These methods can generate new metabolic reactions since it used the concept of generalize enzyme activity based on third-level of enzyme classification, which is recommended by International Union of Pure and Applied Chemistry and International Union of Biochemistry and Molecular Biology (IUPAC-IUBMB) (NC-IUBMB, 1999). Each enzyme is assigned a four-digit (EC i.j.k.l) Enzyme Commission (EC) number. Enzyme-catalyzed reactions based on the third-level enzyme classification, EC i.j.k, are not substrate specific, and thus, these described the transformation of functional groups. With this concept, those methods create rules of enzyme transformation to search for possible reactions catalyzed by

the third-level enzyme. Even those methods are capable to create new metabolic reactions, but the information of gene and protein for further study is unavailable. Thus, the rule-based methods are high risk as predicted chemicals may not be possible in real experiments.

PathMiner, Pathway generation, PathPred, BNICE, and Prioritization methods require background information are categorized as the rules-based methods, while OptStrain (Pharkya et al., 2004), Network expansion (Handorf et al., 2005), DESHARKY (Rodrigo et al., 2008), Graph-based pathway (Dogrusoz et al., 2009), Path finding (Pey et al., 2011), Probabilistic (Yousofshahi et al., 2011), SPABBATS (Flórez et al., 2011), and ArtPathDesign (this thesis) (Chatsurachai et al., 2013) are categorized as the graph-based methods. The graph-based methods are capable to generate pathways of target without using any background information of enzyme activity. The rule-based methods require much time to search for the optimized pathway due to a huge number of possible pathways from rules and atomic balances, while most of graph-based methods are fast to calculate based on reaction stoichiometry. However, the OptStrain, Network expansion, and DESHARKY applied alternative ways/scores to rank and select the optimized pathways. For example, OptStrain used OptKnock (Burgard et al., 2003) algorithm for further improving yield of target by knockout genes in the host network. DESHARKY used earlier experimental data of RNA polymerase activity as a score to find the optimized pathway, which required more calculation time as well as prior information.

Among key features in Table 1.2, most of earlier methods have not yet been developed to provide specific heterologous pathways and heterologous genes for the

host cell in addition to the catalog of nonnative metabolite especially for the target host cell. These features are significant for researchers in order to improve and /or produce the desire of industrial nonnative metabolites, because expression of heterologous pathways may be low/not active. The main reasons are sometimes occurred by heterologous genes show low/no expression in the target host cell since diversity of species and cofactor balances such as NAD⁺/NADH and NADP⁺/NADPH.

Hence, in this thesis I have developed the *in silico* method according to the purposes (Fig. 1.8), to search for feasible heterologous pathways, suggest rational heterologous genes particular for each host, and provide the catalog of nonnative metabolites, which are able to be produced by the host cell. This system was named as ArtPathDesign (Artificial heterologous Pathway Design) and could overcome the problems about host-specific heterologous pathways and genes, as well as the catalog of nonnative metabolites (Table 1.2). In addition, this computational method greatly reduces times and costs for data analysis and also experiments.

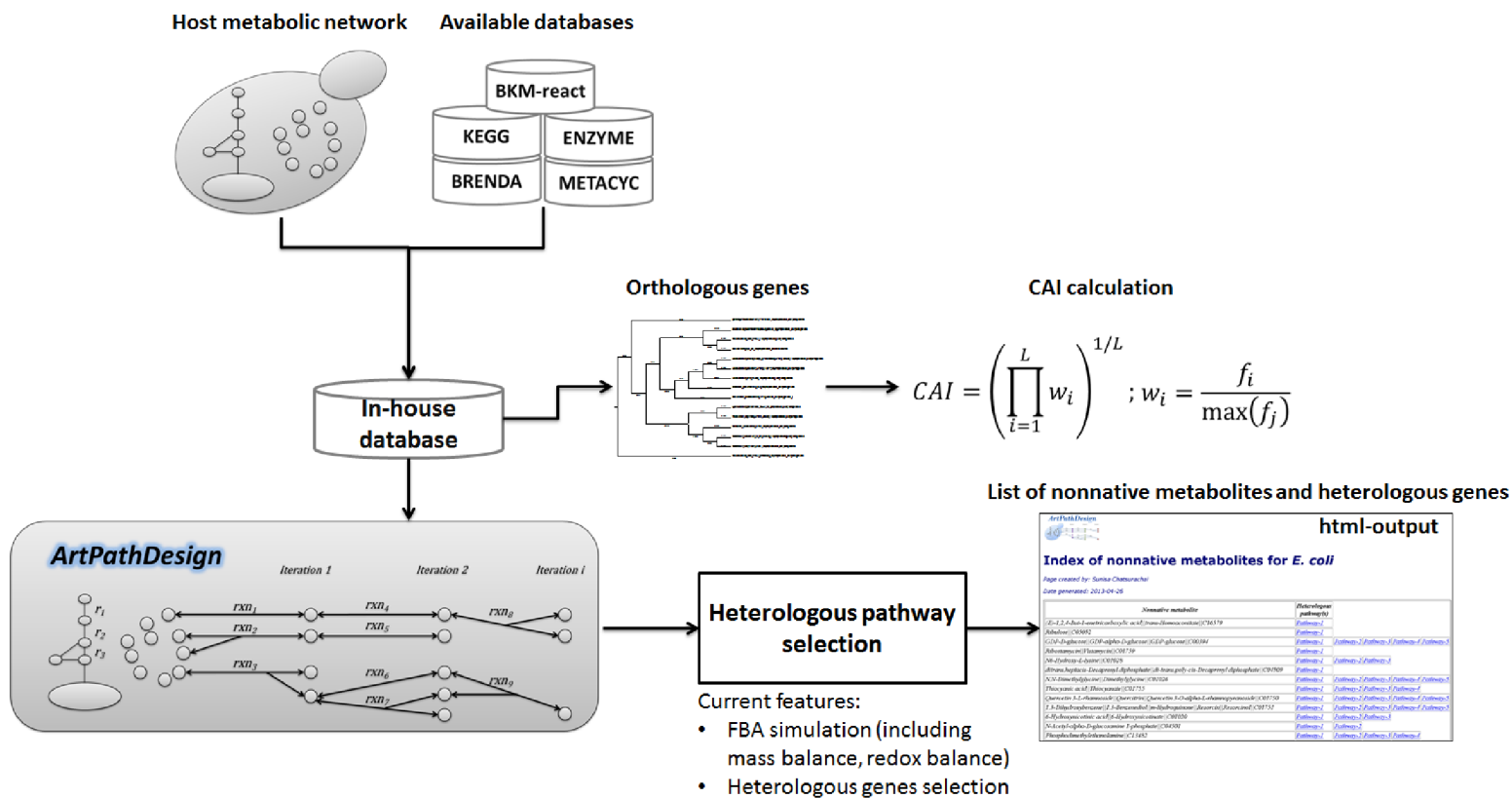


FIG. 1.8 The schematic of the computational method developed in this thesis, named as “ArtPathDesign”.

Host metabolic network and metabolic reactions collected in an in-house database were used as input data for screening heterologous pathways of nonnative metabolites. In parallel, Codon Adaptation Index (CAI) scores of all genes retrieving from the available databases was calculated and used to select candidate genes. FBA simulation was performed to select heterologous pathways. Finally, list of nonnative metabolites and heterologous genes particular for a target host will be listed as the output result in html-format easily opened by web browser such as Google Chrome, Firefox, etc.

Table 1.2 Comparison of computational metabolic pathway search methods

Method name	Reference	Key points of method							
		No requirement of Background	Optimized pathway search (fast)	New pathway	New reaction	Information of new reaction (enzyme/gene)	Host-specific pathway	Host-specific heterologous gene(s)	Catalog of nonnative metabolites (host-specific)
PathMiner	McShan et al. (2003)	✗	✗	✓	✓	✗	✗	✗	✗
Pathways generation	Li et al. (2004)	✗	✗	✓	✓	✗	✗	✗	✗
OptStrain	Pharkya et al. (2004)	✓	✗	✓	✗	✗	✓	✗	✗
Network expansion	Handorf et al. (2005)	✓	✗	✓	✗	✗	✗	✗	✗
DESHARKY	Rodrigo et al. (2008)	✓	✗	✓	✗	✗	✓	✓	✗
Graph-based pathway	Dogrusoz et al. (2009)	✓	✓	✓	✗	✗	✗	✗	✗
PathPred	Moriya et al. (2010)	✗	✗	✓	✓	✗	✗	✗	✗

Table 1.2 (Continued)

Method name	Reference	Key points of method							
		No requirement of Background	Optimized pathway search (fast)	New pathway	New reaction	Information of new reaction (enzyme/gene)	Host-specific pathway	Host-specific heterologous gene(s)	Catalog of nonnative metabolites (host-specific)
BNICE	Finley et al. (2009)	✗	✗	✓	✓	✗	✗	✗	✗
Prioritization	Cho et al. (2010)	✗	✗	✓	✓	✗	✗	✗	✗
Path finding	Pey et al. (2011)	✓	✓	✓	✗	✗	✗	✗	✗
Probabilistic	Yousofshahi et al. (2011)	✓	✓	✓	✗	✗	✗	✗	✗
SPABBATS	Flórez et al. (2011)	✓	✓	✓	✗	✗	✗	✗	✗
ArtPathDesign (this thesis)	Chatsurachai et al. (2013)	✓	✓	✓	✗	✗	✓	✓	✓

1.3 Modeling and analysis of genome-scale metabolic networks

Generally, a model is created to simulate a process or a set of processes observed in the experiment in order to better understand mechanisms of process and to predict outcomes for a given set of specific input parameters. Therefore, to gain insight into cellular metabolism, a genome-scale metabolic network model has been an important tool. The genome-scale metabolic network is reconstructed from genome sequence annotation and biochemical reactions mining from databases (Table 1.3) and literatures. The construction of metabolic network is an iterative decision-making and time-consuming process and it could take up to one month to several months to complete a comprehensive model of a genome-scale metabolic network. In addition, an accurate model could be done by using experimental data retrieved from literatures. *E. coli* (Reed et al., 2003), *S. cerevisiae* (Mo et al., 2009), *B. subtilis* (Oh et al., 2007) and *C. glutamicum* (Shinfuku et al., 2009) models are the examples of accurate models predicting cellular phenotypes under various conditions.

Table 1.3 Database useful for pathway mining and curation

Database	URL
KEGG	http://www.genome.jp/kegg/pathway.html
MetaCyc	http://metacyc.org/
BRENDA	http://www.brenda-enzymes.org/
BKM-react	http://bkm-react.tu-bs.de/
BiGG	http://bigg.ucsd.edu/
CyanoBase	http://genome.microbedb.jp/cyanobase
Biochemical Pathway Maps	http://web.expasy.org/pathways/

Several modeling techniques (Raman et al., 2006) are available to analyze and to simulate cellular mechanism/response such metabolic flow when metabolic pathway or environment altered. Among these techniques, flux balance analysis (FBA) is the most commonly used. FBA can provide estimations of the metabolic fluxes on the genome-scale metabolic network, thereby making it possible to predict the growth rate of an organism or the rate of target production (Orth et al., 2010).

Theory of flux balance analysis (FBA)

FBA assumes that metabolic fluxes will reach a steady state constrained by the stoichiometry (Kauffman et al., 2003). The stoichiometric constraints lead to an underdetermined system; however, a bounded solution space of all feasible fluxes can be identified. This solution space can be further restricted by specifying maximum and minimum fluxes through any particular reaction and by specifying other physiochemical constraints. Here, the metabolic fluxes are estimated by these constraints, and the constraints can be refined by adding experimental data.

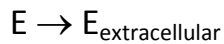
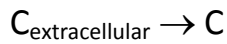
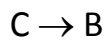
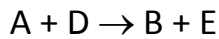
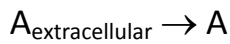
When the solution space that describes the capability of the organism is defined, the metabolic network's behavior can be studied by optimizing the steady-state behavior with respect to some objective function. The simulation results can then be experimentally verified and used to further strengthen the model. Finally, the iterative model refinement procedure can result in predictive models of cellular metabolism (Mo et al., 2009; Oh et al., 2007; Reed et al., 2003; Shinfuku et al., 2009). To better understand how to formulate a FBA problem, the steps are explained in detail and demonstrated through the toy metabolic network (Fig. 1.6).

FBA model formulation contains 4 steps. (Adapted from (Kauffman et al., 2003; Raman and Chandra, 2009))

Step I. System definition

Development of a flux balance model requires the definition of all the metabolic reactions and metabolites. Fig. 1.6 shows the toy metabolic network, which contains 2 intracellular and 5 transport reactions. There are 5 intracellular metabolites (A-E) and 5 extracellular metabolites (A_{ext} , C_{ext} , D_{ext} , E_{ext} , and Biomass).

The biochemical reactions of the network are listed here.



Step II. Mass balance

Once all reactions and transport mechanism of the system are identified, a dynamic mass balance is derived for all intracellular metabolites in the metabolic network shown here (equation 1.1a-e).

$$\frac{d[A]}{dt} = b_1 - v_1 \quad (1.1a)$$

$$\frac{d[B]}{dt} = v_1 + v_2 - b_2 \quad (1.1b)$$

$$\frac{d[C]}{dt} = b_3 - v_2 \quad (1.1c)$$

$$\frac{d[D]}{dt} = b_4 - v_1 \quad (1.1d)$$

$$\frac{d[E]}{dt} = v_1 - b_5 \quad (1.1e)$$

The mass balance is defined in terms of the flux through each reaction and the stoichiometry of that reaction, thus, a set of ordinary differential equations is obtained (equation 1.2a-1.2e). In this analysis, the steady state of the system is assumed, and corresponds to the case that input fluxes are equal to output fluxes, as follows.

$$\frac{d[A]}{dt} = b_1 - v_1 = 0 \quad (1.2a)$$

$$\frac{d[B]}{dt} = v_1 + v_2 - b_2 = 0 \quad (1.2b)$$

$$\frac{d[C]}{dt} = b_3 - v_2 = 0 \quad (1.2c)$$

$$\frac{d[D]}{dt} = b_4 - v_1 = 0 \quad (1.2d)$$

$$\frac{d[E]}{dt} = v_1 - b_5 = 0 \quad (1.2e)$$

The differential equations can be represented using a matrix notation, where " S " is the stoichiometric matrix and " v " is the vector of the fluxes. The goal of FBA is to identify the metabolic fluxes under steady-state condition of the metabolic network.

At steady-state condition, the different equations of all metabolites can display in matrix form shown in Fig. 1.9 below.

Stoichiometric maxtrix

$$\begin{array}{c}
 \begin{array}{ccccc}
 & v_1 & v_2 & b_1 & b_2 & b_3 & b_4 & b_5 \\
 \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}
 \end{array}
 \end{array}$$

S matrix

↓

$$\frac{d}{dt} \begin{pmatrix} A \\ B \\ C \\ D \\ E \end{pmatrix} = \sum_{j=1}^R S_{ij} \cdot v_j \quad \forall i \in M$$

M = a set of metabolites
 R = a set of fluxes (reactions)

↓

Steady state Mass balance

$$\sum_{j=1}^R S_{ij} \cdot \begin{pmatrix} v_1 \\ v_2 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{pmatrix} = 0$$

v vector

FIG. 1.9 Matrix notations of differential equations under steady-state condition

Step III. Defining measurable fluxes and/or range of fluxes

In general, there are more reactions (or fluxes) than the number of metabolites, and thus the steady-state solution for the metabolic fluxes is underdetermined.

Therefore, additional constraints are required to uniquely determine the steady-state flux distribution. One way to get the additional constraints for the metabolic network is measuring metabolic fluxes experimentally. In linear algebra, to solve the problem, the number of variables should be equal to the number of equations, that is, the measurement of fluxes such as substrates uptake, by-products formation rate should be used as additional constraints. For example, the toy metabolic system contains 5 equations and 7 variables; it requires at least 2 measured fluxes of substrate uptake or production rate to find the unique solution of this problem. The exact flux values are commonly not defined, but rather a range of allowable flux values. The ranges of flux values are either retrieved from experiments or literature, which are used as additional constraints.

The examples of additional constraints for the toy metabolic network are demonstrated below. Here, b_1 , b_3 and b_4 represent substrate uptake rates (equation 1.3 h-j) that can be observed from experiments.

$$0 \leq v_1 \leq 10 \quad (1.3a)$$

$$0 \leq v_2 \leq 10 \quad (1.3b)$$

$$0 \leq b_1 \leq 10 \quad (1.3c)$$

$$0 \leq b_2 \leq 10 \quad (1.3d)$$

$$0 \leq b_3 \leq 10 \quad (1.3e)$$

$$0 \leq b_4 \leq 10 \quad (1.3f)$$

$$0 \leq b_5 \leq 10 \quad (1.3g)$$

$$b_1 = 4.2 \quad (1.3h)$$

$$b_3 = 5.5 \quad (1.3i)$$

$$b_4 = 3.2 \quad (1.3j)$$

Step IV. Optimization

A genome-scale metabolic network always has more reactions (fluxes) in the system than the number of metabolites corresponding to the underdetermined system, thus there are allowable solution spaces/flux distributions (Fig. 1.10). To find the unique/optimal solution, an optimization technique is commonly applied for measuring the internal fluxes in the metabolic network. Fig. 1.10 shows the flux distribution of the toy metabolic network predicted by using the optimization technique with the objective function to maximize Biomass production (the flux of b_2).

Using the optimization technique, the metabolic network is assumed to be optimized with respect to a target objective function. This allows the underdetermined system to be formulated as the optimization problem. The objective functions such as maximization of biomass, minimization of substrate uptake, maximization of ATP production, etc. are widely used to estimate cellular metabolisms and to provide predictions which can be verified by experimental data (Mo et al., 2009; Oh et al., 2007; Reed et al., 2003; Schuetz, Kuepfer, & Sauer, 2007; Shinfuku et al., 2009). The optimization problem of the genome-scale metabolic network is formulated below.

Problem:

$$\text{Maximize/Minimize} \quad \sum_j^R c_j \cdot v_j \quad (\text{Problem 1})$$

Subject to.

1. $\sum_j^R S_{ij} \cdot v_j = 0, \forall i \in M \text{ and } \forall j \in R$ (mass balance constraints)
2. Linear inequality or equality constraints (additional constraints)

where,

R is a number of reactions (fluxes) in the system or cell.

M is a number of metabolites in the system or cell.

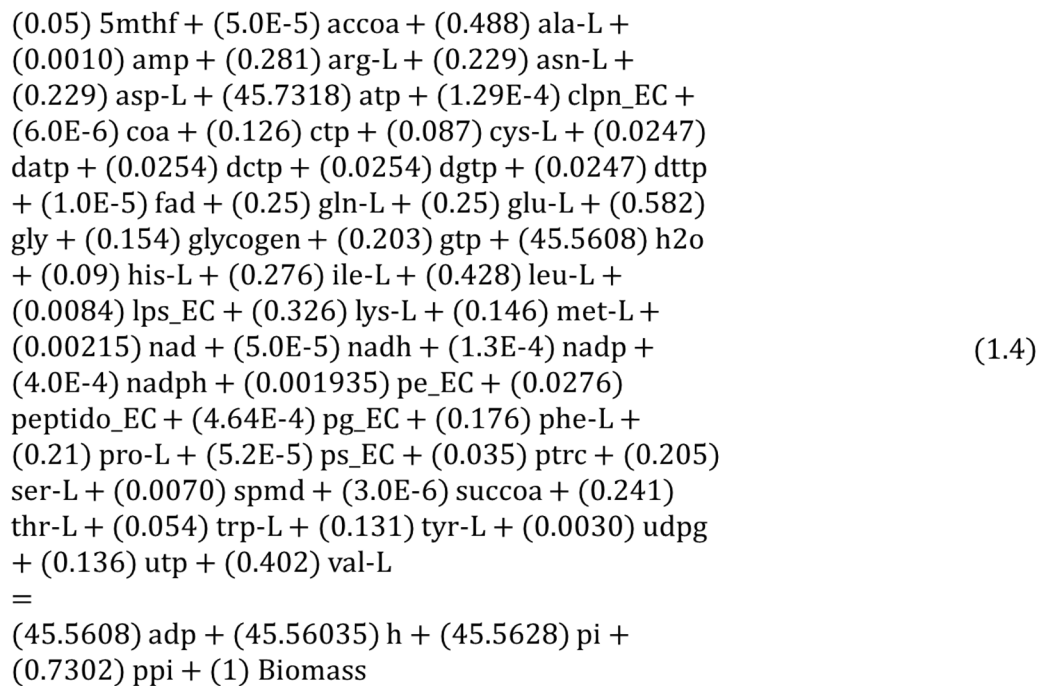
c_j represents weight of the individual flux of the j^{th} reaction that contributed to the objective function.

v_j represents metabolic fluxes of the j^{th} reaction.

S_{ij} represents the stoichiometric coefficient indicating the amount of the i^{th} metabolite produced per unit of flux of the j^{th} reaction.

The aim of FBA is to maximize or minimize the objective function (Problem 1) that is subject to mass balance and additional constraints. The output of this problem is a particular flux distribution of vector, v , which maximizes/minimizes the objective function (Orth et al., 2010).

In general, for the genome-scale metabolic network of the host, the biomass equation is generated based on the ratios of cellular components such as amino acids, RNA, DNA, etc., which are either estimated from experiments and/or genome information. Here is the example of the *E. coli*'s biomass reaction (equation 1.4) retrieved from (Reed et al., 2003):



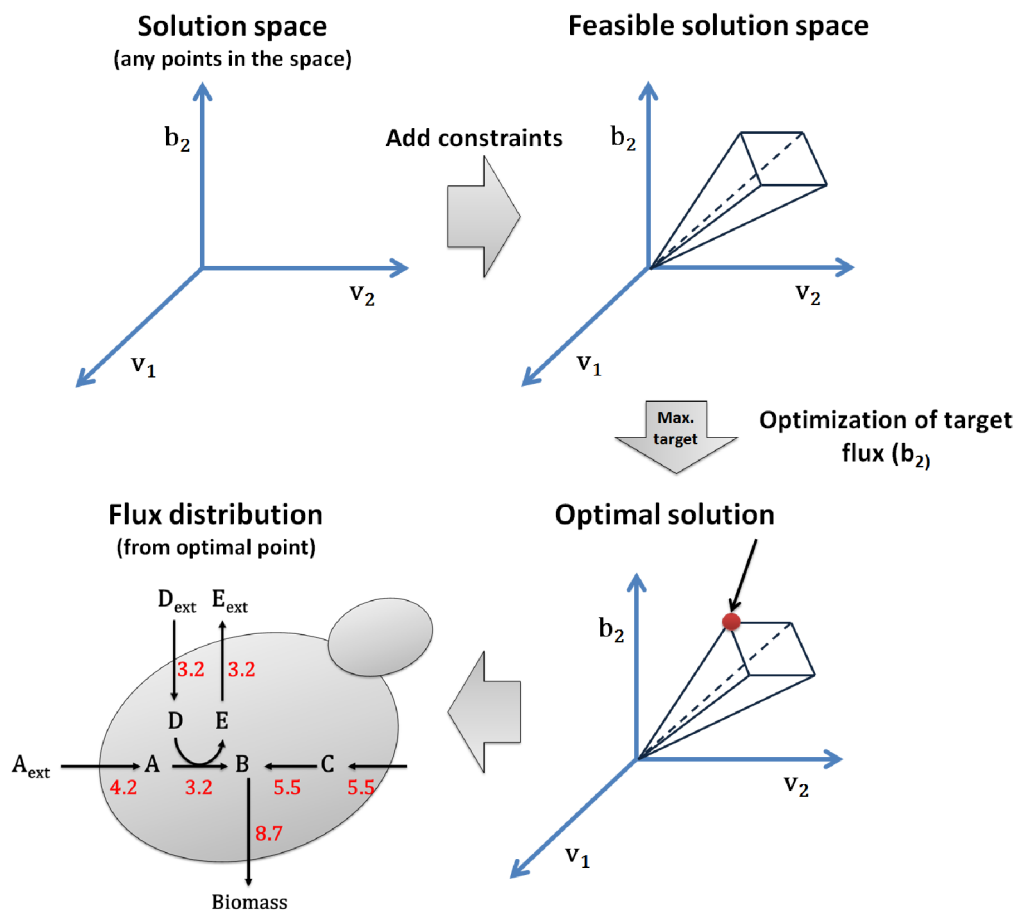


FIG. 1.10 Optimization of the system with the objective function to maximize the flux of b_2 . The b_2 is the biomass production of the toy network (see in Fig. 1.5), which is applied to obtain one optimal solution represented as red dot. The flux distribution of the optimal point is labeled in red on the toy metabolic network map.

1.4 Microorganisms used as industrial cell factories

Since the sequencing of the first complete microbial genome of *Haemophilus influenza* (Fleischmann et al., 1995), a hundred of microbial genomes have been sequenced and archived for public research in GenBank database (Benson et al., 2009). This availability of data provides the scientists to make a genome-scale metabolic model for discovering new information for better understanding cellular properties and processes. In last decade, a genome-scale metabolic model, which is a mathematical model to represent cellular metabolism in linear algebra form, has been used for metabolic engineering by integration of laboratory data such as genome, transcriptome, proteome, metabolome and so forth. Generally, a genome-scale metabolic model was constructed by using genome information such as gene, protein, and metabolic network. As shown in Fig. 1.11, the rapid proliferation of genome sequencing projects over the last decade has resulted in an exponential growth in the amount of genomic DNA sequences and information available for reconstructing genome-scale metabolic models.

The complete genome sequence for a number of microorganisms has been established. Thus, the genome information is available to construct the model that helps for the novel metabolic engineering strategies. Well-characterized hosts would be used as cell factories to yield value-added products because of the availability of genome information and genetic manipulation tools. The examples of these famous host models widely used as cell factories for industrial production are briefly summarized as follows.

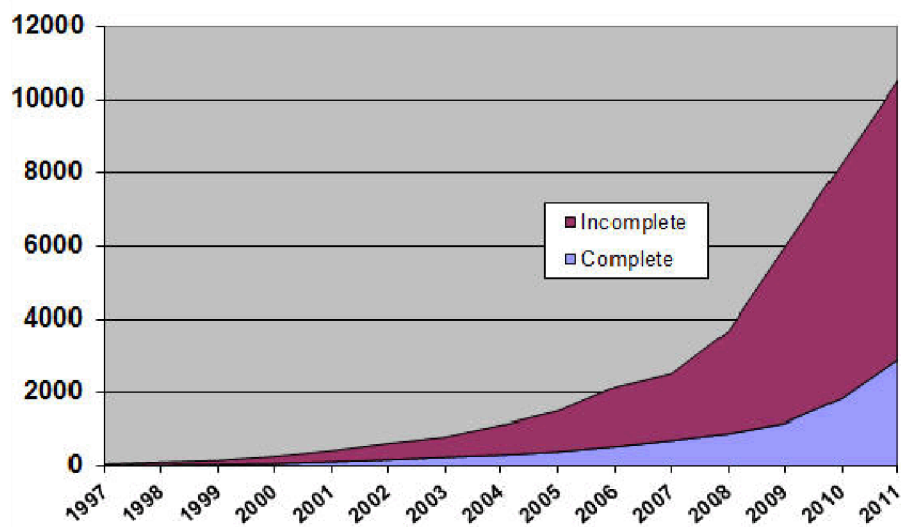


FIG. 1.11 Statistical information of genome projects from GOLD database until October, 2011, Total projects are 10,031 projects (Pagani et al., 2012).

Escherichia coli

E. coli is an aerobic, gram-negative, rod shaped bacteria that can be commonly found in animal feces, lower intestines of mammals, and even on the edge of hot springs. The complete genome sequence of *E. coli* strain K-12 was finished in 1997 (Blattner et al., 1997). Its genome sequence contains about 4.6 Mbps and 4,288 protein-coding genes. The main reasons why *E. coli* becomes famous host for numerous of products are easy for cultivation and fast growth. Currently, *E. coli* has been engineered to produce valuable compounds such as 1,3-propanediol, 1-butanol, lactic and so forth (in Table 1.1). In order to analyze, interpret, and predict cellular behavior, a genome-scale model of *E. coli* was constructed (Reed et al., 2003). This model, named as iJR904, constructed based on *E. coli* K-12 genome annotation data, and showed well predictive simulation comparing with experimental data.

Saccharomyces cerevisiae

A budding yeast *S. cerevisiae* is a eukaryote model for producing alcohols and organic acids such as lactic acid and succinic acid (Table 1.1). The genome sequence of the yeast *S. cerevisiae* was completed in 1996 (Goffeau et al., 1996) and contains about 12 Mbps and defines 5,885 protein-coding genes. Besides, the genome metabolic model of yeast *S. cerevisiae* is also available (Mo et al., 2009). This model named as iMM904 predicted intracellular flux changes consistent with published measurements.

Corynebacterium glutamicum

C. glutamicum, a gram-positive microorganism, is one of the most important bacteria in industrial biotechnology with an annual production of more than 2 million tons of amino acids mainly, L- glutamate, L-serine and L-lysine (Becker and Wittmann, 2012). The *C. glutamicum* genome consists of a single circular chromosome with 3.3 Mbps in size and comprises 3,002 protein-coding genes (Kalinowski et al., 2003). Additionally, its genome-scale metabolic model was constructed and demonstrated metabolic profiles that corresponding with experimental data (Shinfuku et al., 2009).

Bacillus subtilis

B. subtilis, a rod-shaped gram-positive bacterium naturally found in soil and plants, is well-recognized as a producer of enzymes such as proteases and amylases (Zweers et al., 2008). In addition, industrial compound like 2,3-butanediol and isobutanol have also been produced by *B. subtilis* (Biswas et al., 2012; Jia et al., 2012). Its genome is about 4.2 Mbps and comprises 4,100 protein-coding genes (Kunst et al., 1997).

Genome-scale metabolic model of *B. subtilis* was published in 2007 and this *in silico* model could predict growth phenotypes of knock-out strains that found to be quite consistent with experimental observations (Oh et al., 2007).

The microorganisms mentioned above are ideal hosts for bioengineered products such as industrial chemicals, since they exhibit high growth activity under various conditions as well as easy to genetically manipulated (Christina, 2010). Moreover, the genome-scale metabolic models of these microorganisms are available for scientists to use as the tools to identify metabolic engineering strategies such as gene amplification and deletion for strain improvements. For example, target genes to improve lycopene production in *E. coli* were successfully identified using *in silico* simulation and corresponding to enhance the lycopene production in *in vivo* experiment (Choi et al., 2010). Another example is gene knockout simulation to guide target genes for improving L-valine in *E. coli* (Park et al., 2007).

1.5 Objectives of the work

The current demands of fuels and chemical feedstocks are critically increased, while petroleum resources are limited and unsustainable. Moreover, fuels and industrial chemicals by petroleum-based process show negative impacts on environment. The alternative route to produce energy and valuable chemicals using microorganisms becomes an attractive way. However, some microorganisms are not easy to cultivate and produce high level of target products. In addition, the very large amount of possible heterologous pathways is generated without any background in

formation and impossible to handle by human. Therefore, a computational or *in silico* platform to design and select such suitable heterologous pathways is required. Several pathway design methods had been reported (Table 1.2), however, it still lacked of the method that can provide host-specific heterologous pathways, host-specific heterologous genes as well as a catalog of nonnative metabolites particular for each host.

Since the *in silico* platform to design and select suitable heterologous pathways into particular hosts is still not developed, the goal of this thesis is to develop the system aiming to provide necessary information to scientists for producing target metabolites in cell factories hosts such as *B. subtilis*, *C. glutamicum*, *E. coli* and *S. cerevisiae*. The algorithm to screen heterologous pathways for the specific host was developed. This algorithm can provide all possible pathways for the production of nonnative metabolites that are non-existent in the host. Then, parameters used for selection of heterologous genes were applied to select feasible pathways for the production of nonnative metabolites. Thus, a rational heterologous pathway design system named as “ArtPathDesign” (Artificial heterologous Pathways Design) was proposed for an efficient production of nonnative metabolites.

1.6 Outline of the thesis

The thesis consists of 4 chapters, and a schematic outline of the thesis is shown in Fig. 1.12.

Chapter 1 deals with the background and motivation of this thesis. Literature review of metabolic pathway design methods is summarized, and comparison of those methods is demonstrated. The objectives and schematic of this thesis are also described.

Chapter 2 deals with data collection, an in-house database construction and the development of the algorithm for screening heterologous pathways of nonnative metabolites in 3 host cell factories, *E. coli*, *C. glutamicum*, and *S. cerevisiae* as templates. In this chapter, the K_m value is applied in order to select candidate heterologous genes based on enzyme-substrate affinity corresponding to the heterologous pathway. However, the information of K_m value is depending on experimental data and several enzymes show no information of K_m values. Besides, to obtain or improve the production of nonnative metabolites, host-specific heterologous pathways and host-specific heterologous genes are important features. Thus, a new selection parameter could be applied for selection of suitable heterologous pathways and genes.

Chapter 3 deals with the new score that used for selection of candidate heterologous pathways and genes which are specific for each host. Codon Adaptation Index (CAI) is accepted as a measurement of synonymous codon usage bias, which is one of the most important factors effecting on heterologous enzymes expression. Thus, CAI was

applied in this study. With the CAI as the selection score, the host-specific pathways and host-specific genes features are included in the improved *in silico* system, ArtPathDesign. Furthermore, a catalog of nonnative metabolites especially for the target host is improved and available as html file that is well-formed representations.

Chapter 4 deals with the general conclusion and future perspective of this research.

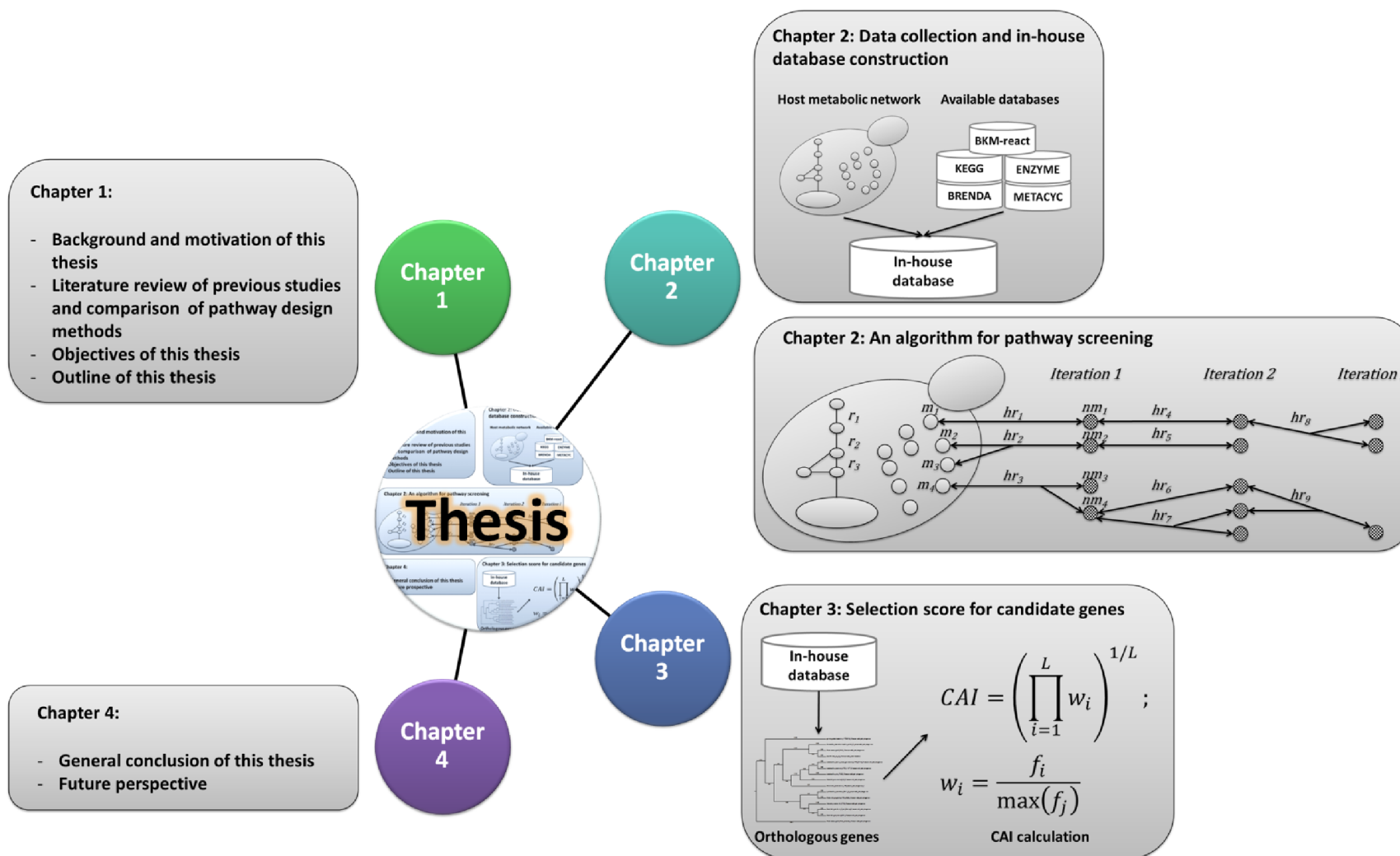


FIG. 1.12 Outline of the thesis

Chapter 2

Development of an algorithm to design heterologous pathway

2.1 Introduction

Recognizing the potential depletion of petroleum resources, researchers have become increasingly interested in production of fuels and industrial chemicals by microorganisms (Dugar and Stephanopoulos, 2011; S. K. Lee et al., 2008; Schneider and Wendisch, 2011). Such biosynthesized materials include fuels, plastics, polymers, solvents and drugs (Becker and Wittmann, 2012; J. W. Lee et al., 2011; Papini et al., 2010; B.-W. Wang et al., 2012). To produce such industrially useful materials, modifications of host's metabolic networks are generally required. Target metabolites are frequently produced by incorporating heterologous metabolic pathways into well-characterized host microorganisms, such as *E. coli*, *S. cerevisiae*, and so on, that I mentioned in previous chapter. However, the selection of suitable heterologous metabolic pathways for host organisms is often difficult due to the complexity of metabolic network. Although copious data on metabolic reactions and enzymes have been available in the literature and available databases such as KEGG, ENZYME, and BRENDA, constructing a target production pathway using a host's metabolic network with satisfying the metabolic balances requires a scientist's experience and intuition. Thus, the development of an appropriate *in silico* platform

can facilitate industry-focused metabolic network design by providing possible heterologous pathways for target metabolite production.

Currently, several *in silico* pathway search methods have been developed and used to produce target metabolites (Cho et al., 2010; Dogrusoz et al., 2009; Finley et al., 2009; Flórez et al., 2011; Handorf et al., 2005; C. Li et al., 2004; McShan et al., 2003; Moriya et al., 2010; Pey et al., 2011; Pharkya et al., 2004; Rodrigo et al., 2008; Yousofshahi et al., 2011). In previous chapter, the comparison of these methods is summarized. Even numerous methods are available for screening heterologous pathways of target metabolites, there still remains a lack of agreement on how to choose heterologous pathways and host microorganisms for target production.

In this chapter, I first developed a novel pathway search algorithm that identifies the shortest pathway between a host's metabolic network and target metabolites when heterologous reactions are added to the host's metabolic network. Using this algorithm, all producible target metabolites listed in databases were screened. In addition, to select candidate heterologous enzymes, K_m value was utilized as a selection score. That is, among the heterologous genes that coding enzyme having minimum K_m value was selected for the construction of heterologous pathways. Then, for all producible target metabolites, the production yields were estimated by using flux balance analysis (FBA), assuming the steady-state conditions and the maximization of target or biomass production rate. By analyzing the entire list of producible target metabolites in several different hosts, a set of rational heterologous pathways and host microorganisms that will likely produce desired targets were selected.

2.2 Methods

2.2.1 Construction of an in-house database of metabolic reactions

All known metabolic reactions were considered as candidate heterologous reactions that could be added to the host metabolic network. First, an in-house database of metabolic reactions was constructed based on data stored in KEGG ligand section (Kanehisa et al., 2008) and BRENDA (Chang et al., 2009) databases. All metabolic reaction information regarding genes, enzymes, pathways, and organisms in the KEGG database was collected into the database, which was developed using PostgreSQL 9.0 (The PostgreSQL Global Development Group). The Michaelis-Menten constants (K_m) of the enzymatic reaction data were retrieved from BRENDA.

2.2.2 Genome-scale metabolic model of host microorganisms

In this chapter, 3 well-characterized and industry-used microorganisms, namely, *E. coli*, *C. glutamicum*, and *S. cerevisiae* were adopted as host microorganisms to be engineered for the target metabolite productions. *E. coli* has been exploited for such industrially valuable compounds as L-phenylalanine, L-tyrosine, 1-butanol, and 1,2-propanediol (Clomburg and Gonzalez, 2011; Juminaga et al., 2012; Shen et al., 2011). *C. glutamicum* is widely used in amino acid production (Becker and Wittmann, 2012). *S. cerevisiae* is an important producer of alcohols and organic acids such as lactate (Hong and Nielsen, 2012). These 3 organisms are widely used for bioengineering since they exhibit high growth activity under several conditions and are easily genetically manipulated

(Christina, 2010). Genome-scale metabolic models of *E. coli* (iJR904)(Reed et al., 2003), *S. cerevisiae* (iMM904) (Mo et al., 2009), and *C. glutamicum* (Shinfuku et al., 2009), based on earlier metabolic constructions with slight modification were used in this study. Because the pathway search algorithm developed in this chapter uses the heterologous reactions listed in the KEGG database, all metabolite IDs in the earlier genome-scale metabolic models were converted to the KEGG compound ID format using metabolite name matching and manually checking.

2.2.3 Heterologous pathway identification for target production

An algorithm to identify heterologous reaction(s) producing a target metabolite within a host microorganism was developed. The algorithm expands the host's metabolic network by sequentially adding heterologous metabolic reactions from the constructed in-house database. The concept of the heterologous pathway identification is shown in Fig. 2.1, and the procedure is as follows:

1. A set of metabolites M_0 and a set of metabolic reactions Rxn_0 are defined as those present in the genome-scale metabolic network of the host microorganism.
2. From the in-house database, heterologous reactions that satisfy the following conditions are collected:
 - i. The reaction does not exist in Rxn_0 , and
 - ii. It can produce metabolites that do not exist in M_0 from a metabolite in M_0 . A set of these heterologous reactions is defined

as Rxn_1 , and a set of metabolites produced by reactions in Rxn_1 is defined as M_1 .

3. In the same way, Rxn_i is the set of reactions not present in $\{Rxn_0, Rxn_1, \dots, Rxn_i\}$ which can produce metabolites not existing in $\{M_0, M_1, \dots, M_i\}$ from metabolites included in those sets. This expansion procedure is iterated until no further reaction is connectable to the expanded metabolic network.

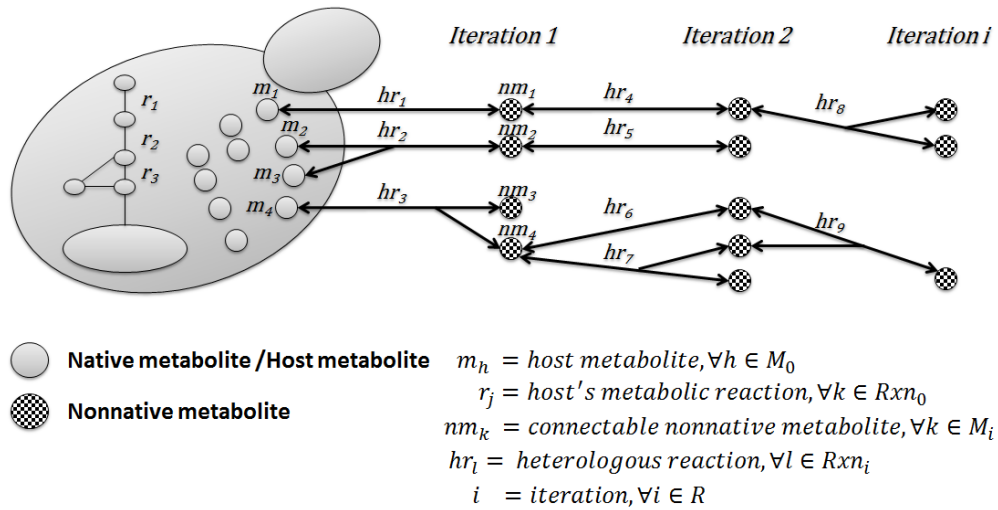


FIG. 2.1 The concept of the algorithm to identify heterogeneous pathways. By sequentially adding heterogeneous reactions, all nonnative metabolites are able to connect to native/host metabolites will be obtained. The process will stop when no further reactions could connect nonnative metabolite to the previous expanded network.

If a target metabolite is included in a nonnative metabolite set M_i , a set of heterogeneous reactions that are necessary to produce the target metabolite can be identified. For simplicity, all metabolic reactions in the database were assumed to be reversible. Of course some reactions are known to be irreversible, such as the carboxylation and decarboxylation reactions classified by

Nomenclature Committee of the international Union of Biochemistry and Molecular Biology (NC-IUBMB, 1999). However, for the majority of reactions in the database, directional information is limited and thus the reversibility of the reactions is difficult to judge. To avoid the risk of missing important heterologous pathways due to misjudgment of their reaction reversibility, all reactions are assumed to be reversible. This strategy here is to initially screen all possible heterologous pathways regardless of reaction irreversibility, then decide whether the predicted pathway is plausible based on physiological knowledge of the reaction irreversibility.

2.2.4 Flux balance analysis (FBA)

FBA is based on a genome-scale metabolic model and optimization of a specific objective flux by linear programming (Kauffman et al., 2003; Orth et al., 2010). FBA was used to estimate the metabolic flux profile of metabolic networks expanded with heterologous reactions. A pseudo steady-state is assumed, that is, the net sum of all production and consumption fluxes for each internal metabolite is zero. In matrix notation, this condition is represented as $S \cdot v = 0$, where S is the stoichiometric matrix representing the stoichiometry of metabolic reactions in the network and v is the vector of metabolic fluxes. In FBA, the flux profile (constrained by steady-state) is determined by optimizing a specific objective function. The biomass production flux is one of several widely used objective functions that can be maximized. The flux profiles obtained by maximizing biomass production fluxes are known to be well correlated with those

obtained experimentally (Mo et al., 2009; Schuetz et al., 2007; Shinfuku et al., 2009).

For simulation, the coefficients of metabolites representing biomass production flux were extracted from earlier studies (Mo et al., 2009; Reed et al., 2003; Shinfuku et al., 2009). Another objective function, the production flux of the target metabolite, was applied to judge whether the target metabolite was producible by the metabolic network. In all of the FBA simulations, glucose was chosen as the sole carbon source and the following external metabolites were allowed to freely transport through the cell membrane: CO₂, H₂O, SO₄ or SO₃, and NH₃. All calculations were performed using MATLAB 2009b (MathWorks Inc., Natick, MA). The linear programming problem was solved using GLPK 4.34 (GNU Linear Programming Kit) via MATLAB interface.

2.3 Results and discussion

2.3.1 Identification of heterologous pathway(s)

Table 2.1 shows information of metabolic reactions and metabolites obtained from 1,139 species were collected from KEGG database and deposited in the constructed in-house database (published as the Additional file 1 of (Chatsurachai et al., 2012)). To screen the target metabolites that are producible by the host microorganisms *S. cerevisiae*, *E. coli*, and *C. glutamicum*, the host's metabolic network was iteratively expanded by adding heterologous metabolic reactions as described in the method section.

Table 2.1 Statistics of the constructed in-house database used in this study

In-house database (version 1.0)	
Source database	KEGG
No. of reactions	7,769
No. of compounds	6,635
No. of reversible reactions	7,769
No. of irreversible reactions	0

Fig. 2.2 displays the number of nonnative metabolites connected to the host's metabolic network as a function of the number of heterologous reactions. Fewer than 33 heterologous reactions are required to connect 3,154, 3,244, and 3,112 nonnative metabolites to the host's metabolic networks of *S. cerevisiae*, *E. coli*, and *C. glutamicum* respectively. The list of metabolites connected to the host's metabolic networks is presented in the Additional files 2, 3, 4 provided at supplementary section of the publication (Chatsurachai et al., 2012). To this list, the K_m values of heterologous enzymes were added. Knowing the K_m assists in deciding which heterologous enzymes originating from various organisms in the BRENDA database displaying minimum K_m of the corresponding heterologous enzymes are also listed, since the enzyme from this organism is expected to have highest affinity among the orthologous enzymes to the corresponding substrate. Importantly, these identified heterologous reactions of nonnative metabolite production agreed well with those widely used in metabolic engineering and which are important to the industry (Table 2.2), such as isoprene, α -farnesene, poly- β -hydroxybutyrate (PHB), and cadaverine.

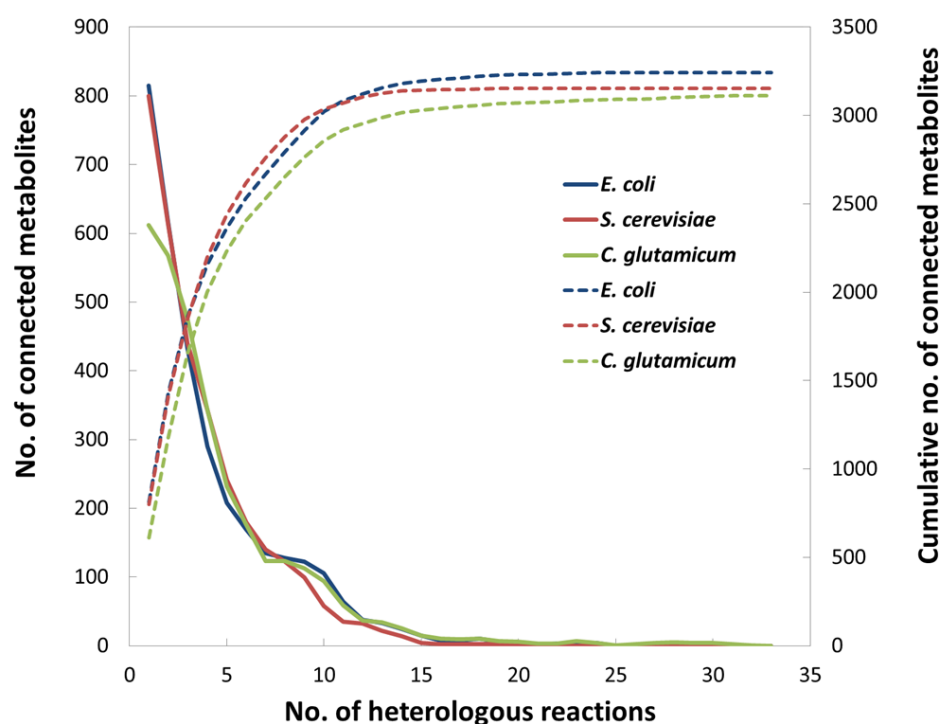


FIG. 2.2 Number of connected nonnative metabolites produced by heterologous reactions in 3 host microorganisms. The first vertical axis (solid line) shows the number of connected metabolites in each iteration, while the second vertical axis (dotted line) shows the cumulative number of the connected metabolites.

As an example, the production pathway of 1,3-propanediol (C02457) by *E. coli* and *S. cerevisiae*, which were adopted in earlier studies (Cameron et al., 1998; Nakamura and Whited, 2003), are shown in Fig. 2.3. In previous studies, C02457 production proceeded via conversion of glycerol to 3-hydroxypropanal using glycerol dehydratase (encoded by *dhaB1-B3*). 1,3-Propanediol was then produced, aided by 1,3-propanediol oxidoreductase (encoded by *dhaT*). In this study, the screened heterologous pathways for C02457 production exactly matched those of earlier studies. In *E. coli*, the screened production pathways of isoprene, α -farnesene, and PHB derived by this algorithm were also identical to those of the

earlier studies, while similar heterologous genes introduced to the alternative hosts (*C. glutamicum* and *S. cerevisiae*) additionally produced these targets (see Table 2.2). Moreover, both reported and alternative production pathways were screened by the proposed algorithm. For instance, It was found that *E. coli* cells can produce (R)-propane-1,2-diol when methylglyoxal reductase and lactaldehyde reductase are added to the metabolic network, which has not been reported so far. Similar alternative pathways were found for the production of itaconate, cis,cis-muconate, 2,3-dihydroxybenzoate, and so forth. These results suggest that the developed algorithm successfully identified the metabolic reactions necessary for the target productions and could assist in screening for heterologous pathways of target productions.

Next, I investigated whether these connectable metabolites are producible from glucose as a sole carbon source, by using FBA simulations. In this simulation, the production flux of each nonnative metabolite was used as an objective function to be maximized under the steady-state assumption. When the maximum production flux of a nonnative metabolite is zero, this metabolite is non-producible under the given condition. The maximum production fluxes of all connectable nonnative metabolites were calculated. 28% of the connectable nonnative metabolites of *E. coli* could not be produced using glucose as a sole carbon source. Similarly, 33% of the connectable nonnative metabolites of *S. cerevisiae* and 16% of the connectable nonnative metabolites of *C. glutamicum* were non-producible under this condition. These metabolites could not be produced since they are disconnected from glycolysis. In *E. coli*, these

metabolites included trans-aconitase (C02341), butyrate (C00246), acetoacetate (C00164), and L-lactaldehyde (C00424).

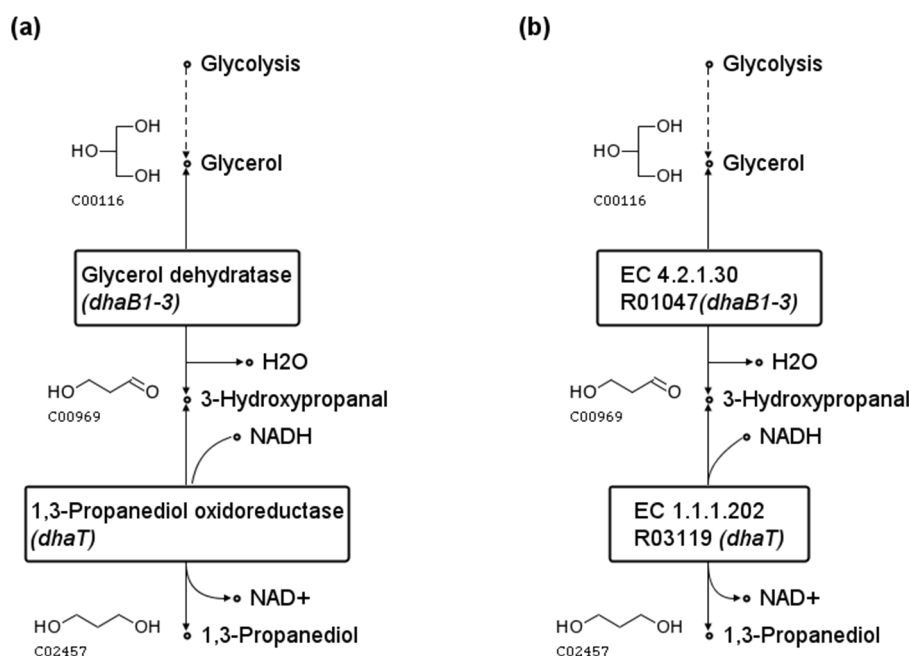


FIG. 2.3 Heterologous pathways for 1,3-propanediol production: (a) the production pathway described in earlier studies, in *E. coli* (Cameron et al., 1998; Nakamura and Whited, 2003); (b) the pathway identified by my algorithm in either *E. coli* or *S. cerevisiae* as the host.

2.3.2 Evaluation of production feasibility

To evaluate the feasibility of nonnative target metabolite production, FBA simulations were performed under conditions of maximizing biomass production following heterologous reaction expansion of the genome-scale metabolic model. Metabolic flux profiles calculated by maximization of biomass production rates have been shown to closely represent those in real microorganisms (Edwards and Palsson, 2000a; Edwards et al., 2001; Feist and Palsson, 2010; Schuetz et al.,

2007; Varma and Palsson, 1994). Such agreement can be explained by the growth optimization of microorganisms through evolutionary dynamics (Fong et al., 2003). Furthermore, for the mutant strains constructed in the laboratory, the cells could achieve the near-optimal metabolic state calculated by the FBA simulation after long-term cultivation (Cornelius et al., 2011; Edwards and Palsson, 2000b; Gerdes et al., 2003; Soyer and Pfeiffer, 2010), via the selection of faster growing cells. Thus, it is expected that if a nonnative target metabolite is produced in the FBA simulation under maximized biomass production, that target may be feasibly manufactured.

In Fig. 2.4, the number of target metabolites produced under maximized biomass production was plot, versus the number of heterologous reactions necessary for metabolite production. A threshold yield (1%) was set to identify the produced metabolites because the production yields of some metabolites were positive but extremely small. Sometimes the FBA simulation was underdetermined under biomass maximization conditions; that is, the solution was not unique. In such cases, following the maximization of biomass production, the production flux of the target metabolites was further maximized with fixing the maximized biomass production, to obtain a unique flux profile that would produce the target. In the simulations, a micro-aerobic condition was used to screen the target metabolites produced under the biomass maximization condition, in which significantly a larger number of target metabolites were produced than under anaerobic conditions, and at the same time all anaerobically produced metabolites were included.

Table 2.3 shows the representative target metabolites produced under biomass maximization, together with their corresponding heterologous reactions. The mechanisms involved in these reactions can be classified into two categories. One is based on the production of oxygen as a by-product of the targets. Since the simulations were performed under the micro-aerobic condition, oxygen supply increased the biomass production by activating the electron transfer system and facilitating adenosine triphosphate production. Therefore, if the heterologous reactions used to produce the target are accompanied by oxygen production, the target can be produced under minimum biomass production flux. For example, pentane-2,4-dione was produced by introducing a single heterologous reaction into *E. coli* and *S. cerevisiae*, whereas two heterologous reactions were necessary to produce this metabolite in *C. glutamicum*. Vanillin can be produced under the same mechanism by introducing 4 heterologous reactions into the *E. coli* and *C. glutamicum* metabolic networks.

Another mechanism is associated with NADH oxidation. Under micro-aerobic condition, the cellular growth of microorganisms can be limited by NAD regeneration, which is necessary for glycolysis activity, and which occurs through NADH oxidization. Thus, when the heterologous reactions producing the targets are associated with NADH oxidization, these heterologous reactions are activated when the biomass production is maximized. This phenomenon occurs, for example, in the production of (R)-propane-1,2-diol and 2-propyn-1-ol.

It is found that some metabolites are produced only by *E. coli* under conditions of maximum biomass production, such as (R)-propane-1,2-diol and

adipate semialdehyde. Unlike *S. cerevisiae* and *C. glutamicum*, *E. coli* possesses NAD transhydrogenase, which can convert NADP and NADH to NADPH and NAD respectively (and vice versa). In *E. coli* cells, the excess NADH is converted to NADPH which can then enter the target production pathway.

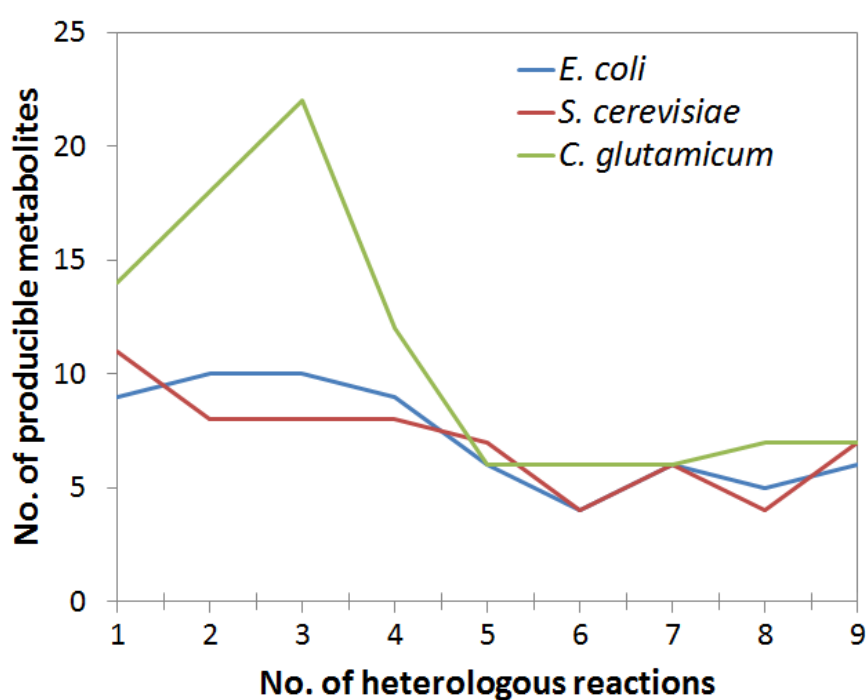


FIG. 2.4 The number of metabolites producible under biomass maximization conditions with the addition of < 10 heterologous reactions.

Table 2.2 Examples of nonnative metabolites for which my algorithm detected heterologous reactions matching those of previous studies

Compound (synonym separated by a semicolon)	KEGG ID	Heterologous reaction(s) from the literature	Reference	Evaluation of <i>in silico</i> design
Isoprene; 2-methyl-1,3-butadiene	C16521	Introduced ispS gene from <i>Populus nigra</i> to <i>Escherichia coli</i>	(Zhao et al., 2011)	Identical reaction found in <i>E. coli</i> and in <i>Saccharomyces cerevisiae</i> and <i>Cerevisiae glutamicum</i> as the host
α -Farnesene	C09665	Introduced farnesene synthase from plant to <i>E. coli</i>	(C. Wang et al., 2011)	Identical reaction found in <i>E. coli</i> and in <i>S. cerevisiae</i> and <i>C. glutamicum</i> as the host
Poly- β -hydroxybutyrate; PHB	C06143	Introduced phbC and phbB from <i>Streptomyces aureofaciens</i> to <i>E. coli</i>	(Mahishi et al., 2003)	Identical reaction found in <i>E. coli</i> and in <i>S. cerevisiae</i> and <i>C. glutamicum</i> as the host
Cadaverine; 1,5-pentanediamine; 1,5-diaminopentane	C01672	Introduced ldcC from <i>E. coli</i> to <i>C. glutamicum</i>	(Becker and Wittmann, 2012; Kind et al., 2010)	Identical reaction found in <i>C. glutamicum</i> and in <i>S. cerevisiae</i> as the host
Amorpha-4,11-diene	C16028	Introduced AMS1 from the plant <i>Artemisia annua L.</i> to <i>E. coli</i>	(Lindahl et al., 2006; Wallaart et al., 2001)	Identical reaction found in <i>E. coli</i> and <i>S. cerevisiae</i> and in <i>C. glutamicum</i> as the host
Propane-1,3-diol; 1,3propanediol; trimethylene glycol	C02457	Introduced glycerol dehydratase and 1,3- propanediol oxidoreductase from <i>Klebsiella pneumoniae</i> to <i>E. coli</i> .	(Cameron et al., 1998; Nakamura and Whited, 2003)	Identical reaction found in <i>E. coli</i> and in <i>S. cerevisiae</i> as the host
Ethanol; ethyl alcohol; methylcarbinol	C00469	Introduced pyruvate decarboxylase and alcohol dehydrogenase genes from <i>Zymomonas mobilis</i> to <i>C. glutamicum</i>	(Inui et al., 2004)	Identical reaction found in <i>C. glutamicum</i> as the host
(R,R)-Butane-2,3-diol; (R,R)-2,3-Butanediol; (R,R)-2,3-Butylene glycol	C03044	Introduced acetolactate decarboxylase and butanediol dehydrogenase genes to <i>E. coli</i>	(Nielsen et al., 2010)	Identical reaction found in <i>E. coli</i> as the host

Table 2.2 (Continued)

(R)-Propane-1,2-diol; (R)-1,2-propanediol; (R)-propylene glycol	C02912	Introduced glycerol dehydrogenase gene from <i>Klebsiella pneumoniae</i> and used aldehyde dehydrogenase to produce product in <i>E. coli</i>	(Altaras and Cameron, 1999)	Alternative pathway found to produce target by adding methylglyoxal reductase and lactaldehyde reductase to <i>E. coli</i>
		Introduced glycerol dehydrogenase and methylglyoxal synthase genes from <i>E. coli</i> to <i>S. cerevisiae</i>	(W. Lee and Dasilva, 2006)	Alternative pathway found to produce target by adding methylglyoxal reductase and lactaldehyde reductase to <i>S. cerevisiae</i>
Itaconate; itaconic acid; methylenesuccinic acid	C00490	No information	No information	EC 4.2.1.4-citrate dehydratase and EC 4.1.1.6-aconitate decarboxylase were found to be added to <i>E. coli</i> as the host.
<i>cis,cis</i> -Muconate; <i>cis,cis</i> -hexadienedioate; <i>cis,cis</i> -2,4-hexadienedioic acid	C02480	Introduced aroZ, aroY, and catA to <i>E. coli</i>	(Niu et al., 2002)	Alternative pathways from anthranilate or 2,3-dihydroxybenzoate to produce catechol, which is a substrate for <i>cis,cis</i> -muconate production
Adipate; adipic acid; hexanedioate; hexan-1,6-dicarboxylate	C06104	Introduced aroZ, aroY, and catA to <i>E. coli</i> for producing <i>cis,cis</i> -muconate and then convert to adipic acid by chemical synthesis	(Niu et al., 2002)	Alternative pathway found to produce the target by adding 5 heterologous reactions to <i>E. coli</i> or <i>C. glutamicum</i> as the hosts (see Additional files 4 and 5 in (Chatsurachai et al., 2012) for enzyme information)

Table 2.3 Examples of producible nonnative metabolites under conditions of maximized biomass production

Nonnative metabolites	Host network	By-product	No. of reaction(s)	Heterologous reaction(s)	EC number
Pentane-2,4-dione	<i>E. coli</i> , <i>S. cerevisiae</i>	Oxygen	1	Pentane-2,4-dione + oxygen \leftrightarrow acetate + methylglyoxal	1.13.11.50
	<i>C. glutamicum</i>	Oxygen	2	Glycerone phosphate \leftrightarrow methylglyoxal + orthophosphate Pentane-2,4-dione + oxygen \leftrightarrow acetate + methylglyoxal	4.2.3.3 1.13.11.50
Vanillin (4-hydroxy-3-methoxy-benzaldehyde)	<i>E. coli</i> , <i>C. glutamicum</i>	Oxygen, NADH	4	Formaldehyde + NAD ⁺ + H ₂ O \leftrightarrow formate + NADH + H ⁺	1.2.1.46
				3-Dehydroshikimate \leftrightarrow 3,4-dihydroxybenzoate + H ₂ O	4.2.1.118
				Vanillate + oxygen + NADH + H ⁺ \leftrightarrow 3,4-dihydroxybenzoate + NAD ⁺ + H ₂ O + formaldehyde	1.14.13.82
				Vanillate + NAD ⁺ + H ₂ O \leftrightarrow 4-hydroxy-3-methoxy-benzaldehyde + oxygen + NADH + H ⁺	1.2.3.9
(R)-Propane-1,2-diol	<i>E. coli</i>	NAD ⁺	2	(R)-Lactaldehyde + NAD ⁺ + H ₂ O \leftrightarrow (R)-lactate + NADH + H ⁺	1.2.1.23
				(R)-Propane-1,2-diol + NAD ⁺ \leftrightarrow (R)-lactaldehyde + NADH + H ⁺	1.1.1.77
2-Propyn-1-al	<i>S. cerevisiae</i>	NAD ⁺	3	3-Oxopropanoate \leftrightarrow acetaldehyde + CO ₂	4.1.1.-
				3-Oxopropanoate \leftrightarrow propynoate + H ₂ O	4.2.1.27
				2-Propyn-1-al + NAD ⁺ + H ₂ O \leftrightarrow propynoate + NADH + H ⁺	1.2.1.3

Table 2.3 (Continued)

Adipate semialdehyde	<i>E. coli</i>	NADP+	6	Succinyl-CoA + acetyl-CoA \leftrightarrow CoA + 3-oxoadipyl-CoA	2.3.1.174
				(3S)-3-Hydroxyadipyl-CoA + NAD ⁺ \leftrightarrow 3-Oxoadipyl-CoA + NADH + H ⁺	1.1.1.35
				5-Carboxy-2-pentenoyl-CoA + H ₂ O \leftrightarrow (3S)-3-hydroxyadipyl-CoA	4.2.1.17
				Adipyl-CoA + FAD \leftrightarrow 5-carboxy-2-pentenoyl-CoA + FADH ₂	1.3.99.-
				Adipate + CoA + ATP \leftrightarrow Adipyl-CoA + AMP + diphosphate	6.2.1.-
				Adipate semialdehyde + NADP ⁺ + H ₂ O \leftrightarrow adipate + NADPH + H ⁺	1.2.1.4

2.3.3 Differences in target production capacity among host microorganisms

While screening for heterologous pathways to produce the target metabolites discussed earlier, differences in production capacity between the 3 host microorganisms emerged; for example, a group of metabolites was producible by the addition of heterologous reactions to one of the hosts, but was not produced by the other hosts. To characterize the differences in target production capacity, the producible metabolites (the Additional files 5, 6, 7 of supplementary of the publication (Chatsurachai et al., 2012)) were categorized using the KEGG Orthology database (Kanehisa et al., 2008). A chi-square statistical analysis was then performed to identify the categories in which the frequency of producible metabolites is significantly higher than expected. Fig. 2.5 shows the 10 categories that demonstrated significant differences ($P < 0.001$). As shown in the figure, metabolites belonging to 5 categories, namely, “tyrosine metabolism”, “dioxin degradation”, “benzoate degradation”, “chlorocyclohexane and chlorobenzene degradation”, and “xylene degradation”, tended to be producible by *S. cerevisiae* and *C. glutamicum*, but were non-producible in *E. coli* cells.

Similarly, the metabolites in “flavonoid biosynthesis”, “phenylpropanoid biosynthesis”, and “nicotinate and nicotinamide metabolism” were preferentially generated by *E. coli* and *C. glutamicum*. Metabolites assigned to “porphyrin and chlorophyll metabolism” also tended to be produced in *C. glutamicum* cells. Likewise, the metabolites assigned to “biosynthesis of 12-, 14-, and 16-membered macrolides” were produced preferentially in *E. coli* cells. Such differences in production capabilities result from the different metabolic

pathways by which the hosts produce necessary substrates, and from cellular compartmentalization in the yeast strain (which is absent in the bacterial strains).

In yeast cells, the compartments present barriers to metabolite transport. For instance, mitochondrial/cytoplasmic interfaces prohibit the production of certain target metabolites when sugar is used as a carbon source. Similarly, the production of metabolites in the “flavonoid biosynthesis” category was inhibited in yeast cells because the transportation of 4-coumarate between the mitochondria and the cytosol is not permitted; therefore, the yeast strain could not produce p-coumaroyl-CoA (required for making chalconoid, an important ingredient in flavonoid biosynthesis). The genome-scale metabolic model used in this study does not account for transportation capabilities between compartments, which are currently unclear for many metabolites, and which might influence the production capacities of target metabolites in real cell systems.

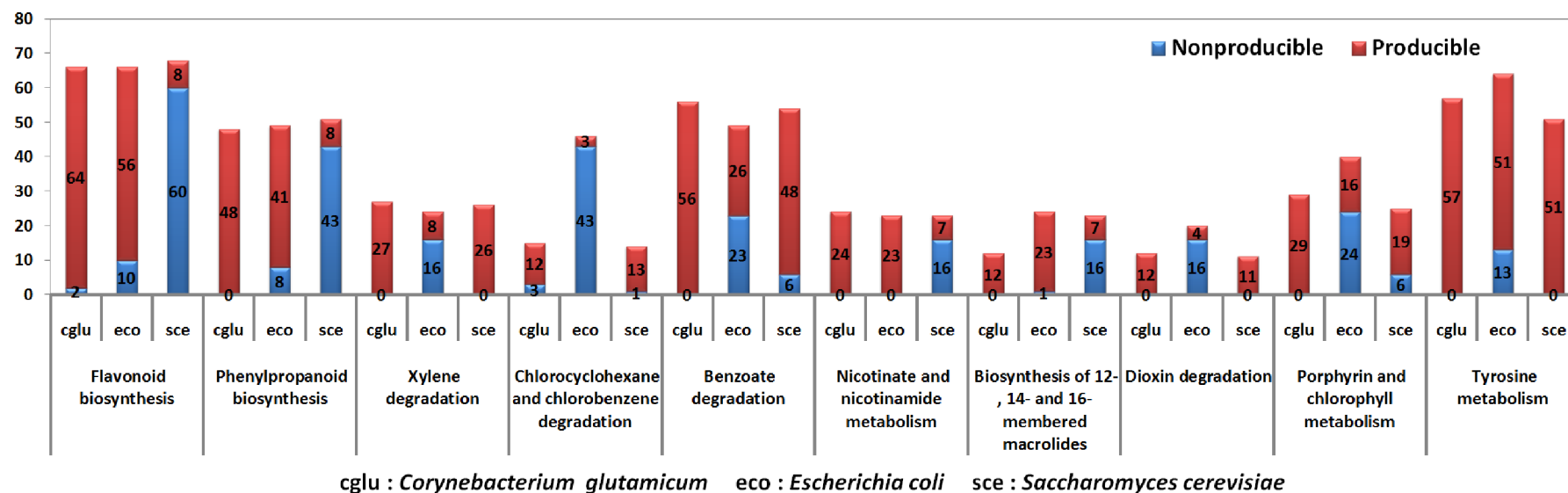


FIG. 2.5 The number of producible and non-producible metabolites in functional categories that exhibit significant differences between host microorganisms. The blue and red bars represent the non-produced and produced metabolites respectively, under conditions of maximized target production.

2.4 Summary

In this chapter, the *in silico* pathway search algorithm for target production was developed, in which iterative additions of heterologous metabolic reactions to host's metabolic networks enable target productions. Biosynthetic capabilities are also evaluated by pathway design and metabolic flux calculations. The 3 industrial host cell factories, *E. coli*, *S. cerevisiae*, and *C. glutamicum* were used as templates. The screened heterologous pathways by using the developed algorithm were validated. The results were consistent with earlier reports. In addition, alternative heterologous pathways that are no reports so far were also suggested, including the production pathways of itaconate, cis,cis-muconate, adipic acid, and so on. Since these compounds are important in industrial chemicals (Table 1.1), these alternative pathways could be options of metabolic engineering strategies in order to produce/improve industrial metabolites by bio-based process.

The computational platform developed in this chapter has included the features which are host-specific heterologous pathways and selection of heterologous genes by using K_m value and provides a catalog of nonnative metabolites including industrial chemicals for specific host cell. However, K_m values are generally obtained from experiments and many enzymes have no information on K_m in the database. Furthermore, in some cases, the heterologous genes are not expressed or low expressed, which can result low production rate of target metabolites. In addition, the catalog of nonnative metabolites in this chapter is available as a table in Microsoft® Excel, which is difficult to search for the information of heterologous genes.

In chapter 3, I will deal with these problems for improving the selection scores of heterologous genes particular for each host cell and improving the catalog of nonnative metabolites into well-format to provide heterologous genes and compounds information.

Chapter 3

Selection of heterologous genes using CAI score

3.1 Introduction

The production of industrial compounds using microorganisms as cell factories has presently become attractive due to the potential depletion of petroleum resources. Metabolic engineering to incorporate heterologous pathways to host cell factories is one major way to produce and improve target chemicals and fuels via bio-fermentation process. In chapter 2, the development of the computational platform are presented, which enables to screen heterologous pathways of nonnative metabolites and used K_m as a parameter to select heterologous genes corresponding to heterologous pathways. Still, a large number of enzymes has no information on K_m values, thus a new and efficient parameter is desirable to select candidate heterologous genes from among numerous of orthologous enzymes with similar enzymatic activities. In addition, the inefficient target production by the introduced heterologous genes could be caused by a low or lack of expression of heterologous enzymes in the host cell. To overcome this problem, heterologous enzymes which have the potential to be highly expressed should be chosen from the screened candidates. It has been established that several factors namely, genome GC content, codon usage, and mRNA secondary structures influence the expression of heterologous enzymes. Amongst these factors, codon usage is known to cause a

relatively high impact on the enzymatic expression levels. In fact, it is demonstrated there is the bias of codon used in high and low expressed genes in several organisms (Gupta et al., 2004; G. Liu et al., 2010; Sharp et al., 1986). In addition, Botzman and Margalit discovered that the global extent of codon usage bias of an organism plays a crucial role in the adaptation of prokaryotes to their environments (Botzman and Margalit, 2011). A few species such as *E. coli* (M.-S. Lee et al., 2011) and *S. cerevisiae* (Norkiene and Gedvilaite, 2012) exhibited enhanced heterologous protein expressions when the codons usage of heterologous genes were replaced with a set of more suitable host codons. Several methods for analyzing codon usage bias have been developed in order to study molecular evolution and heterologous gene/protein expression (Ingvarsson, 2008; Olivares-Hernández et al., 2011; W. Qian et al., 2012; Sharp and Li, 1987; Tao et al., 2009). Codon Adaptation Index (CAI) (Sharp and Li, 1987) is one of the most widely used to estimate the extent of codon usage bias in genes (Martín-Galiano et al., 2004; Nayak, 2012) and proteins (Futcher et al., 1999; Washburn et al., 2001) based on their expression levels. Previous reports employed CAI for optimizing DNA vaccines (Mani et al., 2011), enhancing and altering exogenous and endogenous protein expressions (W. Li et al., 2011; Redemann et al., 2011). These studies suggested that CAI can be used for the selection of heterologous enzymes which is expected to be highly expressed in the host microorganisms.

In this chapter, the computational platform for heterologous pathway search is expanded by incorporating the CAI measure of heterologous genes to select appropriate heterologous enzymes, whose introduction into host microorganisms

would enable efficient target metabolite production. Furthermore, I design the optimized gene sequences for such enzymes by substituting the most frequent codons found in host's highly expressed genes. By integrating the simple *in silico* screening platform (for identifying feasible heterologous pathways developed in chapter 2) with the selected heterologous genes as well as the optimized gene sequences, based on a target host's preferable codons, a rational heterologous pathway design system named "ArtPathDesign" (Artificial heterologous Pathway Design) is proposed for an efficient production of nonnative metabolites. The ArtPathDesign system applied to 3 hosts *E. coli*, *S. cerevisiae* and *B. subtilis* which are recognized as industrial host producers. Using this system, all producible nonnative metabolites of *E. coli*, *S. cerevisiae*, and *B. subtilis* were obtained along with specific information regarding the feasible heterologous enzymes. Furthermore, the catalog of nonnative metabolites which can be produced by each host is improved from Microsoft® Excel-format to Hyper Text Markup Language (HTML)-format that is a well-demonstration of heterologous pathway, gene, and compound information.

3.2 Materials and methods

3.2.1 Constructing an in-house database of metabolic reactions

All known metabolic reactions were considered as candidate heterologous reactions that could be added to the host's metabolic network. An in-house database of metabolic reactions was firstly constructed from data stored in BKM-react (Lang et al., 2011), which is an integrated database resulting from a

combination of database namely, BRENDA (Chang et al., 2009), KEGG (Kanehisa et al., 2008), and MetaCyc (Caspi et al., 2008). This in-house database is the update version of previously constructed that used metabolic reactions from retrieved only from KEGG database. All metabolic reaction information regarding genes, enzymes, pathways, and organisms in the KEGG and the EMBL (Kulikova et al., 2004) databases were collected into the new version of the in-house database developed using PostgreSQL 9.0 (The PostgreSQL Global Development Group).

3.2.2 Screening for artificial heterologous pathways involved in the production of targets

Genome-scale metabolic models of *E. coli* (iJR904)(Reed et al., 2003), *S. cerevisiae* (iMM904) (Mo et al., 2009), and *B. subtilis* (Oh et al., 2007), were used based on earlier metabolic reconstructions with slight modifications. In order to incorporate known metabolic pathways listed in BKM-react database, all metabolite IDs in the earlier genome-scale metabolic models were converted to the KEGG compound ID and MetaCyc compound ID format using metabolic name matching using python scripts and manually search. In this study, the algorithm presented in chapter 2 was applied in order to search heterologous reaction(s) that produce target metabolites within a host microorganism.

3.2.3 Codon Adaptation Index (CAI)

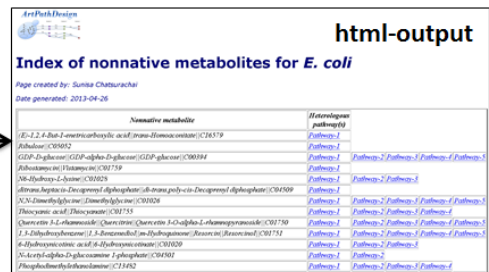
CAI was proposed by Sharp and Li as a measure of synonymous codon bias calculated based on the codon preference of highly expressed proteins, such as ribosomal proteins, and elongation factors (Sharp and Li, 1987). CAI score of a

gene is a measure of favorable codons or optimal codons, which are frequently used by the highly expressed proteins in a given genome, which takes a value from 0 to 1. CAI for a gene with L-amino acids is formulated as follows:

$$CAI = \left(\prod_{i=1}^L w_i \right)^{1/L}$$

$$w_i = \frac{f_i}{\max(f_j)}$$

where, f_i denotes the frequency of i^{th} codon among other synonymous codons coding the same amino acids in the set of highly expressing genes. The relative adaptiveness w_i is defined by the ratio between the frequency of i^{th} codon and that of the most frequently used codon for the amino acid. CAI of a protein is defined as the geometric mean of the relative adaptiveness of all the codons in the coding sequence. For the calculation of CAI, a set of highly expressed genes in the hosts, *E. coli*, *S. cerevisiae*, and *B. subtilis* was collected from literature (Das et al., 2009) and database (Puigbò et al., 2008). The w_i table was created for each host by using sequence of highly expressed genes in the corresponding host strain (see in Table B.1 in Appendix B). The CAI score of orthologous genes of all organisms in KEGG and EMBL database was then calculated. The overall strategy of the computational platform integrating a new selection score, named as ArtPathDesign, developed in this thesis is summarized in Fig. 3.1.



Host metabolic network and metabolic reactions collected in the in-house database were used as input data for screening heterologous pathways of nonnative metabolites. In parallel, CAI scores of all genes retrieving from the available databases was calculated and used to select candidate genes. Finally, the catalog of nonnative metabolites will be generated and available in html-format.

3.3 Results and discussion

3.3.1 Identification of heterologous pathways

Statistics of the updated version of the in-house database is shown in Table 3.1. The updated in-house database contains 17,488 known metabolic reactions, 17,617 metabolites of 2,056 organisms retrieved from BKM-react (shown in Table S1 of supplementary data). All supplementary data (Table S1-S4) in this chapter are available on the web site below.

http://www-shimizu.ist.osaka-u.ac.jp/supplementaryData_artpathdesign.zip

Necessary information regarding genes, enzymes, nucleotide sequences, organisms as well as the reversibility data from KEGG, BRENDA, and EMBL databases were parsed and stored in a version 2.0 of the in-house database.

Table 3.1 Statistics of the in-house database version 1.0 (chapter 2) and version 2.0 constructed in this chapter

	In-house database (version 1.0)	In-house database (version 2.0)
Source database	KEGG	BKM-react
No. of reactions	7,769	17,488
No. of reactions (no EC number)	1,274	0
No. of compounds	6,635	17,617
No. of reversible reactions	7,769	17,488

To archive nonnative production in *E. coli*, *S. cerevisiae*, and *B. subtilis* as the hosts, the algorithm developed in chapter 2, was used to screen heterologous

pathways from the metabolic reactions collected in the in-house database. In brief, by employing this algorithm the host's metabolic network were iteratively expanded by adding heterologous reactions step-by-step in order to link nonnative metabolites. The host's metabolic network expanded until no further nonnative metabolites could connect to it. Fewer than 32 heterologous reactions were required to connect 3,226, 3,298, and 3,211 nonnative metabolites to the host's metabolic networks of *E. coli*, *S. cerevisiae*, and *B. subtilis* respectively. Comparison of results between version 1.0 and 2.0 of the in-house databases are shown in Table 3.2. From this table, it was found in all hosts excluding *E. coli*, the update version of the in-house database was improved as for the number of connectable metabolites. The total number of connectable metabolites when *E. coli* as the host decreased from 3,244 to 3,226 because, removing metabolic reactions, which are no information of EC number, previously used to connect to *E. coli*'s metabolic network in chapter 2. Those metabolites included D-Ribulose (C00309), Hyaluronate (C00518), etc. If no EC number information, it is not possible to select heterologous genes. Therefore, the in-house database version 2.0 contains only metabolic reactions that have information of EC number. The list of metabolites connected to the host's metabolic networks by using the new version in-house database is shown in Table S2 of supplementary data.

Table 3.2 Comparison of total connectable nonnative metabolites between 2 versions of the in-house databases

Host cell	In-house database (version 1.0)	In-house database (version 2.0)
<i>E. coli</i>	3,244	3,226
<i>S. cerevisiae</i>	3,154	3,298
<i>C. glutamicum</i>	3,112	3,321
<i>B. subtilis</i>	3,063	3,211

3.3.2 Relationship between CAI score and protein abundance

To analyze the relationship between CAI scores and protein expression levels, protein abundance data were obtained from PaxDb (M. Wang et al., 2012), which is a repository containing information from different proteome experiments of several organisms. 1,560 coding gene sequences (CDSs) of *E. coli* were obtained from KEGG database. Likewise, 5,004 CDSs of *S. cerevisiae* and protein abundance were collected for correlation analysis. CAI scores of genes were then computed and compared them with the protein abundance data as shown in Fig. 3.2. A statistically significant correlation was observed between CAI scores and log (protein abundance values) (correlation coefficient $r=0.60$ in *E. coli* and $r=0.65$ in *S. cerevisiae*). The observed positive correlation of CAI and protein abundances suggested that the CAI could be used as the index representing potential protein expression levels in host organisms.

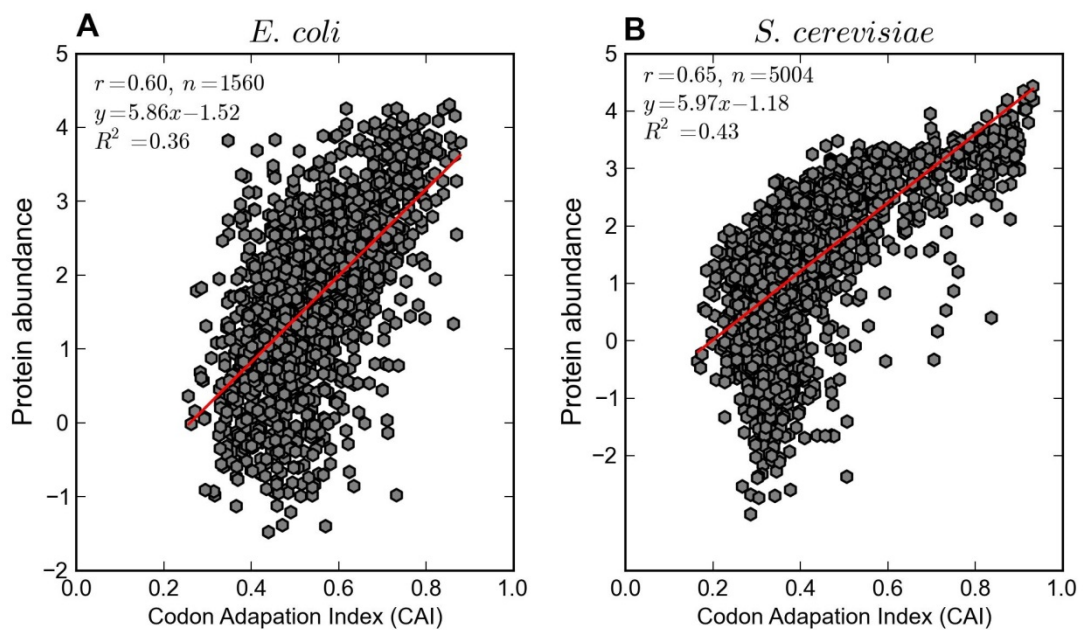


FIG. 3.2 The relationship between the normalized log protein abundance data (ppm: part per million) and CAI scores in (A) *E. coli* (B) *S. cerevisiae*. CAI scores of 1,560 genes in *E. coli* and 5,004 genes in *S. cerevisiae* from PaxDb (Protein Abundance Across Organisms) database are shown. Protein abundance in “ppm” essentially refers to each protein with reference to the complete proteome expression. In other words, it implies the amount of each protein relative to all other protein molecules present in the sample.

3.3.3 Screening of heterologous genes with higher CAI scores

The expression levels of heterologous enzymes are often low (Kleber-Janke and Becker, 2000; Lakey et al., 2000), probably due to the differences in the codon usage between the source and the host organisms. However, based on the positive correlation obtained between CAI scores and protein abundance values (Fig. 3.2), it can be inferred that, amongst the multiple candidate heterologous enzymes involved in the production of a target metabolite, selecting the one

whose corresponding gene sequence has a higher CAI score would result in an enhanced expression of the heterologous enzyme.

In this chapter, potentially highly expressed genes that could enhance the expression level of the corresponding encoded enzymes were identified, and thereby result in a higher productivity of the target metabolites. The process involved determining the CAI scores for all the screened candidate heterologous genes which could lead to the production of all possible target metabolites, and subsequently selecting the candidate heterologous genes with higher CAI scores. Expectedly, for all host microorganisms used in this study, the distributions of CAI scores of the native genes were found to be higher when compared to those of the heterologous genes from other organisms (Fig. 3.3).

It was reported that human genes selected to express in *E. coli* as the host showed low expression (Dai et al., 2013; Gvritishvili et al., 2010; Q. Wang et al., 2012). Thus, I collected the wide type coding sequences of human kallistatin, human Pigment epithelium-derived factor (PEDF), and human cystatin from the previous reports, and calculated CAI scores of these genes. As results, CAI scores of the 3 human genes selected to be expressed in *E. coli* are 0.37, 0.32, and 0.45, respectively. These data suggests that the human genes show low expression due to low CAI scores. In addition, these previous reports also successfully improved the production of proteins by replacing the human genes with the codons that suitable for host *E. coli* without changing amino acids.

In Fig. 3.3, the dashed line represents the distribution of CAI scores of different native genes that originally exist in the host, while the solid line

corresponds to the CAI scores of all possible heterologous genes which could be utilized for the production of the target metabolite. The dotted line represents the CAI scores of the selected heterologous genes. It was observed that the highest CAI scores correspond to the set of heterologous genes which encode enzymes having same functions. As evident, a significant increase in CAI scores was achieved by repeated selection process, thus demonstrating that this approach can overcome the problem of low expression of heterologous enzymes which occur due to differences in the codon usage. The selected heterologous genes required for the production of all possible target metabolites are presented in Table S3 of supplementary data.

Fig. 3.4 illustrates 2 different heterologous pathways where the higher CAI score selection technique was applied for choosing suitable heterologous genes. The production of 1,3-propanediol by *S. cerevisiae* (Rao et al., 2008) is generally achieved by introducing *dhaB* (glycerol dehydratase) from *Klebsiella pneumoniae* and *yqhD* (an alcohol dehydrogenase possessing the activity of 1,3-propanediol dehydrogenase) from *E. coli*. As shown in Fig. 3.4A, the CAI scores of *dhaB* and *yqhD* in *S. cerevisiae* is relatively low in comparison with the native genes, which might result in low expression levels of these heterologous genes. However, this screening platform demonstrated that the productivity of 1,3-propanediol by the host *S. cerevisiae* could be enhanced by introduction of *dhaB* and *dhaT* from *Clostridium perfringens* and *Lactobacillus reuteri*, respectively instead of *dhaB* and *yqhD* from *K. pneumoniae* and *E. coli*, respectively since they have relatively high CAI scores. Another classical example is the heterologous pathway for (R,R)-2,3-butanediol in a host *E. coli*, which has been shown to utilize heterologous

genes from *B. subtilis* (Yan et al., 2009). In this case, the *in silico* system suggested that *alsS* from *Pseudomonas putida*, *alsD* from *Enterobacter cloacae* and *bdh* from *Klebsiella oxytoca* were comparatively better in terms of CAI, shown in Fig. 3.4B. Although other enzymatic characteristics such as K_m value could influence the production of target metabolites, the expression level of a particular heterologous enzyme is crucial for the activation of a specific heterologous pathway in order to achieve a higher productivity of the desired target. Therefore, these findings highlight that the selection of heterologous genes with higher CAI scores is an effective approach to enhance the productivity of the target metabolites.

In addition, the distributions in Fig. 3.3 reveal that in spite of selecting heterologous genes with the highest CAI scores (shown in dotted line) from many organisms, some heterologous genes still exhibit relatively low CAI scores which could result in a low expression level due to the mismatch of codon usage. One possible strategy to overcome this is to calculate an optimized gene sequence encoding a specific heterologous enzyme which has preferable codon usage in the host microorganism (i.e., CAI = 1), and subsequently synthesize this artificial gene experimentally for its introduction into the host. With the growing availability of low-cost de novo gene synthesis (e.g., \$300/1kb in 2012), such optimized gene sequences could be easily employed for bio-production. Fig. 3.5 shows the example of optimized gene sequence comparing with the original gene sequence encoding for catechol 1,2-dioxygenase (*catA*). The optimized sequences encoding all the heterologous enzymes that are required for the production of target metabolites in the 3 host microorganism are given in Table S4 of

supplementary data. Collectively, these data suggest that artificial gene sequences when considered in combination with optimal CAI are valuable, only when the artificial gene synthesis technique is applied to the expansion of the host's metabolic network for the production of targets.

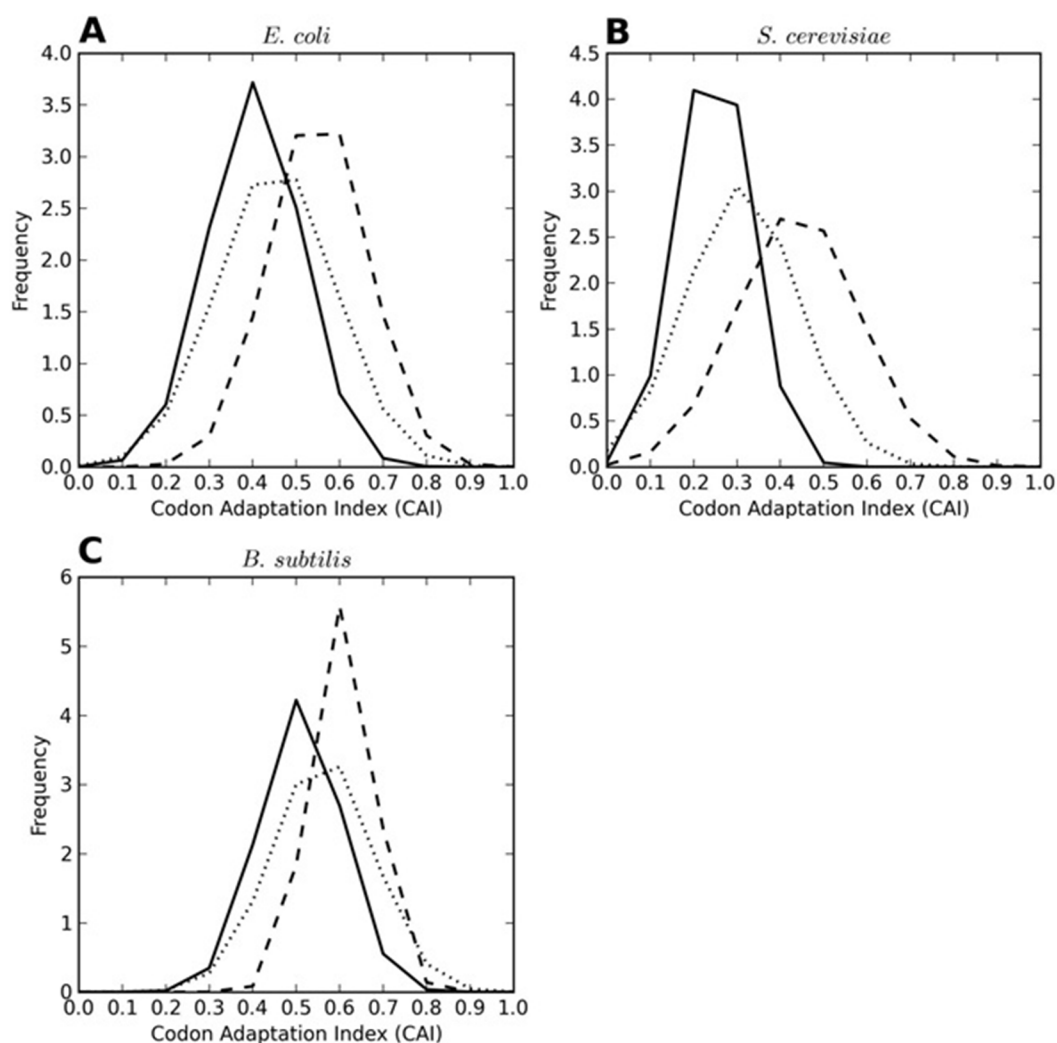


FIG. 3.3 Distribution of CAI scores for each organism (A) *E. coli*, (B) *S. cerevisiae*, and (C) *B. subtilis* as the hosts. Solid line denotes the CAI score distribution of all possible heterologous genes that can be used for nonnative metabolite productions. Dotted line represents the CAI score distribution of all selected heterologous genes with the highest CAI score encoding enzymes having the

same enzymatic function. Dashed line denotes the CAI score distribution of native genes originally from the host microorganism.

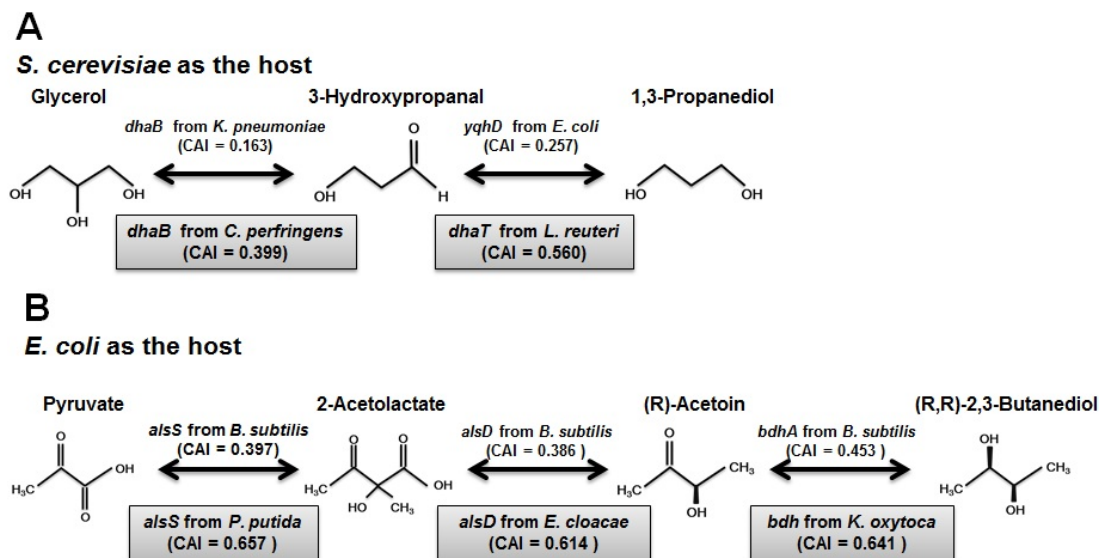


FIG. 3.4 (A) Heterologous pathway for the production of 1,3-propanediol in *S. cerevisiae* (Rao et al., 2008) as well as CAI scores of orthologous genes. The heterologous genes from different organism with the higher CAI scores (shown in box) were selected by employing the improved computational system. (B) Heterologous pathway for the production of (R,R)-2,3-butanediol in *E. coli* (Yan et al., 2009) as well as CAI scores of orthologous gens. The heterologous genes from different organism with the higher CAI scores (shown in box) were selected by employing the improved system. *alsD* encodes acetolactate decarboxylase, *alsS* encodes acetolactate synthase, and *bdh* encodes butanediol dehydrogenase.

Psest_2692_org	atgactgtaaaaatctccaaaccagcgacgttcagaacttcttc [45]
Psest_2692_opt	atgaccgttaaaaatctctcagaccctctgacgttcagaacttcttc [45]
Psest_2692_org	aagggaagccagcggcttcggcaacgacgcggcagcagccgcatg [90]
Psest_2692_opt	aaagaagcgtctgggtttcggtaacgacgcgggttcttctcgtatg [90]
Psest_2692_org	aagaccgtgatcaaccgcattcttggtcgacactgcgaaaaatcgtc [135]
Psest_2692_opt	aaaaccgttatcaaccgtatcctgggtgacaccgcgaaaaatcgtt [135]
Psest_2692_org	gaagacctggaaatcaccaggacgaattctggaaagccgtggac [180]
Psest_2692_opt	gaagacctggaaatcaccaggacgaattctggaaagcgggttgac [180]
Psest_2692_org	tatctcaaccgcctgggcggctcgccacgaagccgggtctgctgggtt [225]
Psest_2692_opt	tacctgaaccgtctgggtggctcgccacgaagcgggtctgctgggtt [225]
Psest_2692_org	gccggcctgggcctggagcacttcctcgacctgctggaagacgcc [270]
Psest_2692_opt	gcgggtctgggtctggaacacttcctggacctgctggaagacgcg [270]

FIG. 3.5 The *catA* sequence (ORF ID: Psest_2692) from *Pseudomonas stutzeri* RCH2 was replaced with the host's favorable codons, for example codon "act" was replace with "acc" coding for threonine. The GC content of original sequence is 64.65%, while GC content of optimized sequence is 57.64%.

Finally, FBA was performed to investigate whether all connected nonnative metabolites are producible by using glucose as a sole carbon source. The maximum fluxes of all connectable nonnative metabolites corresponding to each host were calculated. It was observed that 11% of the connectable nonnative metabolites of *E. coli* could not be produced using glucose as the sole carbon source. Likewise, 27% and 11% of connectable nonnative metabolites of *S. cerevisiae* and *B. subtilis* respectively could not be produced under this condition.

3.3.4 Examples of results in hyper-text based user interface

HTML (Hyper Text Markup Language) is a language for describing web pages, thus outputs in html-format easily open by using web browser such as Google Chrome and Firefox. Python script function was created in order to provide

information of heterologous pathways of nonnative metabolites in html-based format. The hyper-text is automatically generated by these programs and output results are summarized in the file named “index.html” found in archive file available at: <http://www-shimizu.ist.osaka-u.ac.jp/APD.zip>

In order to search for target metabolite, the index.html can be opened via web browser for example Google Chrome as shown in Fig. 3.6. In this figure, how to find heterologous pathway(s) for the production of 1,3-propanediol using search function (Ctrl+F) of Google Chrome is shown. By typing compound name or ID, the possible pathways will be found as shown in Step 3 of Fig. 3.6. By clicking target pathway (for example Pathway-1), the detail of the heterologous pathway will be displayed as shown in Step 4 of Fig. 3.6. Additional example of (R,R)-2,3-butanediol is shown in Fig. 3.7. By using this computational system, the list of all possible metabolites able to connect to host’s metabolic network will be provided as a catalogue for user to search for target production as well as necessary information such as candidate genes, open reading frame ID, and organisms.

Step 1. Open “index.html”

The index.html can open via Google Chrome



Index of nonnative metabolites for *E. coli*

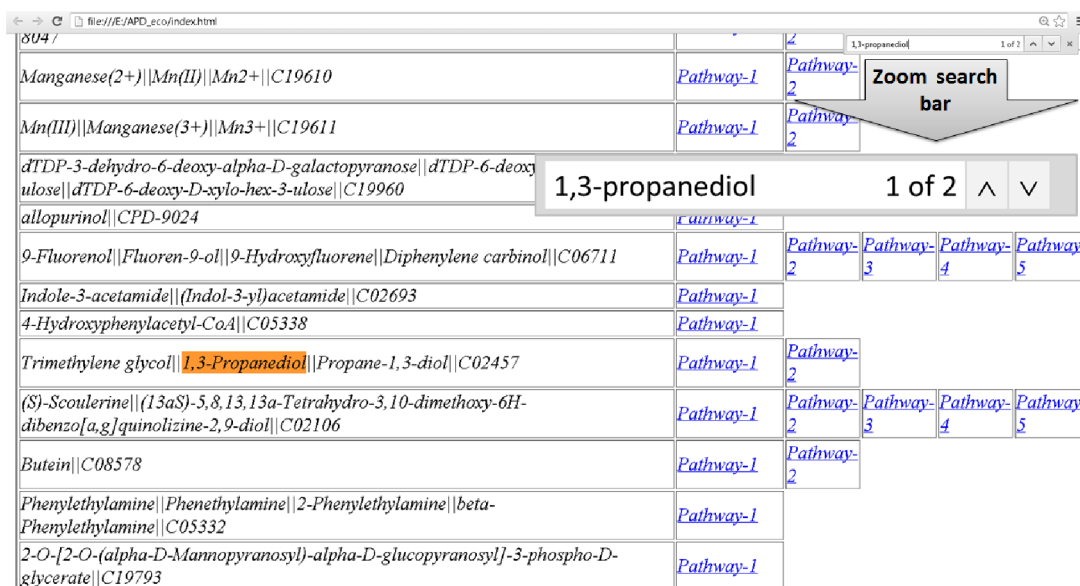
Page created by: Sunisa Chatsurachai

Date generated: 2013-06-07

Nonnative metabolite	Heterologous pathway(s)				
(E)-1,2,4-But-1-enetricarboxylic acid trans-Homoaconitate C16579	Pathway-1				
Ribulose C05052	Pathway-1				
GDP-D-glucose GDP-alpha-D-glucose GDP-glucose C00394	Pathway-1	Pathway-2	Pathway-3	Pathway-4	Pathway-5
N6-Hydroxy-L-lysine C01028	Pathway-1	Pathway-2	Pathway-3		
ditrans,heptakis-Decaprenyl diphosphate di-trans,poly-cis-Decaprenyl diphosphate C04509	Pathway-1				
N,N-Dimethylglycine Dimethylglycine C01026	Pathway-1	Pathway-2	Pathway-3	Pathway-4	Pathway-5
Thiocyanic acid Thiocyanate C01755	Pathway-1	Pathway-2	Pathway-3		

FIG. 3.6 The output result of heterologous pathways for the production of 1,3-propanediol in *E. coli* is provided in html-format. To search for target metabolite, 4 steps are required and shown here. Step 1 shows “index.html”, which is the summarized results of ArtPathDesign (the html file can be easily opened using Google Chrome and Firefox). In this example, the index.html was open by using Google Chrome. Step 2 shows how to search target metabolite using Ctrl+F (search function of Google Chrome) and type compound name or compound ID of target. Step 3 shows the result of target search and two heterologous pathways are available for 1,3-propanediol. Step 4 shows result page of the first candidate pathway in the new page including a structure of compound and detail information of heterologous pathway that are reactions, genes, open reading frame ID, and organisms.

Step 2. Press Ctrl+F (search function of Google Chrome) and Type 1,3-propanediol



Search Results	Pathway-1	Pathway-2	Pathway-3	Pathway-4	Pathway-5
Manganese(2+) Mn(II) Mn2+ C19610	Pathway-1	Pathway-2			
Mn(III) Manganese(3+) Mn3+ C19611	Pathway-1	Pathway-2			
dTDP-3-dehydro-6-deoxy-alpha-D-galactopyranose dTDP-6-deoxy-uloose dTDP-6-deoxy-D-xylo-hex-3-uloose C19960	Pathway-1				
allopurinol CPD-9024	Pathway-1				
9-Fluoreno Fluoren-9-ol 9-Hydroxyfluorene Diphenylene carbinol C06711	Pathway-1	Pathway-2	Pathway-3	Pathway-4	Pathway-5
Indole-3-acetamide (Indol-3-yl)acetamide C02693	Pathway-1				
4-Hydroxyphenylacetyl-CoA C05338	Pathway-1				
Trimethylene glycol 1,3-Propanediol Propane-1,3-diol C02457	Pathway-1	Pathway-2			
(S)-Scoulerine (13aS)-5,8,13,13a-Tetrahydro-3,10-dimethoxy-6H-dibenzo[a,g]quinolizine-2,9-diol C02106	Pathway-1	Pathway-2	Pathway-3	Pathway-4	Pathway-5
Butein C08578	Pathway-1	Pathway-2			
Phenylethylamine Phenethylamine 2-Phenylethylamine beta-Phenylethylamine C05332	Pathway-1				
2-O-[2-O-(alpha-D-Mannopyranosyl)-alpha-D-glucopyranosyl]-3-phospho-D-glycerate C19793	Pathway-1				

Step 3. Red box shows 2 heterologous pathways for 1,3 propanediol in *E. coli*.



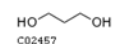
Search Results	Pathway-1	Pathway-2	Pathway-3	Pathway-4	Pathway-5
Manganese(2+) Mn(II) Mn2+ C19610	Pathway-1	Pathway-2			
Mn(III) Manganese(3+) Mn3+ C19611	Pathway-1	Pathway-2			
dTDP-3-dehydro-6-deoxy-alpha-D-galactopyranose dTDP-6-deoxy-D-xylo-hexos-3-uloose dTDP-6-deoxy-D-xylo-hex-3-uloose C19960	Pathway-1				
allopurinol CPD-9024	Pathway-1				
9-Fluoreno Fluoren-9-ol 9-Hydroxyfluorene Diphenylene carbinol C06711	Pathway-1	Pathway-2	Pathway-3	Pathway-4	Pathway-5
Indole-3-acetamide (Indol-3-yl)acetamide C02693	Pathway-1				
4-Hydroxyphenylacetyl-CoA C05338	Pathway-1				
Trimethylene glycol 1,3-Propanediol Propane-1,3-diol C02457	Pathway-1	Pathway-2			
(S)-Scoulerine (13aS)-5,8,13,13a-Tetrahydro-3,10-dimethoxy-6H-dibenzo[a,g]quinolizine-2,9-diol C02106	Pathway-1	Pathway-2	Pathway-3	Pathway-4	Pathway-5
Butein C08578	Pathway-1	Pathway-2			
Phenylethylamine Phenethylamine 2-Phenylethylamine beta-Phenylethylamine C05332	Pathway-1				
2-O-[2-O-(alpha-D-Mannopyranosyl)-alpha-D-glucopyranosyl]-3-phospho-D-glycerate C19793	Pathway-1				

FIG. 3.6 (Continued)

Step 4. Click Pathway-1, the result page will display.



Heterologous Pathway for C02457 in *E. coli* (Glucose as a carbon source)



Compound
structure

Compound ID
(KEGG/MetaCyc format)

Compound name
and synonym

Page created by: Sunisa Chatsurachai

Date: 2013-06-06

Target compound ID	C02457
Target compound Name	Trimethylene glycol 1,3-Propanediol Propane-1,3-diol
List of Heterologous reaction(s)	R03606 R04382
Detail of heterologous reaction	<p>R03606 propane-1,3-diol + NAD+ <=> 3-hydroxypropanal + NADH + H+ 1 C02457 + 1 C00003 <=> 1 C00969 + 1 C00004 + 1 C00080</p> <p>Enzyme information: 1.1.1.202 Gene symbol: dhaT from database: KEGG 1st-candidate gene(s): CAI= 0.633,KPHS_46210,kpm,Klebsiella pneumoniae subsp. pneumoniae HS11286 1,3-propanediol dehydrogenase 2nd-candidate gene(s): CAI= 0.629,KPK_0624,kpe,Klebsiella pneumoniae 342 1,3-propanediol dehydrogenase 3rd-candidate gene(s): CAI= 0.628,KPN_03491,kpn,Klebsiella pneumoniae subsp. pneumoniae MGH 78578 1,3-propanediol dehydrogenase</p>
Detail of heterologous reaction	<p>R04382 Glycerol <=> 3-Hydroxypropanal + H₂O 1 C00116 <=> 1 C00969 + 1 C00001</p> <p>Enzyme information: 4.2.1.30 Gene symbol: dhaB,dhbE,K06122,dhbC from database: KEGG 1st-candidate gene(s): CAI= 0.506,KPN_03487,KPN_03488,KPN_03489,kpn,Klebsiella pneumoniae subsp. pneumoniae MGH 78578 glycerol dehydratase small subunit,glycerol dehydratase large subunit ,glycerol dehydratase medium subunit 2nd-candidate gene(s): CAI= 0.506,KPHS_46170,KPHS_46180,KPHS_46190,kpm,Klebsiella pneumoniae subsp. pneumoniae HS11286 glycerol dehydratase small subunit,glycerol dehydratase large subunit ,glycerol dehydratase medium subunit 3rd-candidate gene(s): CAI= 0.504333333333,KP1_4780,KP1_4781,KP1_4782,kpu,Klebsiella pneumoniae NTUH-K2044 glycerol dehydratase small subunit,glycerol dehydratase large subunit ,glycerol dehydratase medium subunit</p>

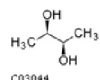
Heterologous
reaction(s) ID

Information of
heterologous
reaction(s)

FIG. 3.6 (Continued)



Heterologous Pathway for C03044 in *E. coli* (Glucose as a carbon source)



Page created by: Sunisa Chatsurachai

Date: 2013-06-07

Target compound ID	C03044
Target compound Name	(R,R)-Butane-2,3-diol[(R,R)-2,3-Butanediol]((R,R)-2,3-Butylene glycol
List of Heterologous reaction(s)	R07477 R01565 R10025
Detail of heterologous reaction	<p>R07477 (R,R)-butane-2,3-diol + NAD+ <=> (R)-acetoin + NADH + H+ 1 C03044 + 1 C00003 <=> 1 C00810 + 1 C00004 + 1 C00080</p> <p>Enzyme information: 1.1.1.4 Gene symbol: butB;butA;EDH from database: KEGG 1st-candidate gene(s): CAI= 0.647,ES15_1123,csk,Cronobacter sakazakii ES15 (R,R)-butanediol dehydrogenase;diacetyl reductase,(R,R)-butanediol dehydrogenase;diacetyl reductase 2nd-candidate gene(s): CAI= 0.6405,KOX_01940,KOX_22375,kox,Klebsiella oxytoca KCTC 1686 (R,R)-butanediol dehydrogenase;diacetyl reductase,(R,R)-butanediol dehydrogenase;diacetyl reductase 3rd-candidate gene(s): CAI= 0.64,CSSP291_04195,csz,Cronobacter sakazakii Sp291 (R,R)-butanediol dehydrogenase;diacetyl reductase,(R,R)-butanediol dehydrogenase;diacetyl reductase</p>
Detail of heterologous reaction	<p>R01565 2-Acetolactate <=> (R)-Acetoin + CO2 1 C00900 <=> 1 C00810 + 1 C00011</p> <p>Enzyme information: 4.1.1.5 Gene symbol: alsD,E4.1.1.5 from database: KEGG 1st-candidate gene(s): CAI= 0.614,ECL_03127,enc,Enterobacter cloacae subsp. cloacae ATCC 13047 acetolactate decarboxylase 2nd-candidate gene(s): CAI= 0.596,APA_42C_03800,apw,Acetobacter pasteurianus IFO 3283-01-42C acetolactate decarboxylase 3rd-candidate gene(s): CAI= 0.596,APA_32_03800,apz,Acetobacter pasteurianus IFO 3283-32 acetolactate decarboxylase</p>
Detail of heterologous reaction	<p>R10025 2 pyruvate <=> 2-acetolactate + CO2 2 C00022 <=> 1 C00900 + 1 C00011</p> <p>Enzyme information: 2.2.1.6 Gene symbol: E2.2.1.6S,ilvM,ilvN,ilvI,ilvH,ilvG,E2.2.1.6L,ilvE from database: KEGG 1st-candidate gene(s): CAI= 0.6564,PFS_2311,PFS_2642,PFS_4257,PFS_4522,PFS_4523,ppf,Pseudomonas putida S16 acetolactate synthase I/II/III large subunit ;acetolactate synthase II small subunit;acetolactate synthase I/III small subunit 2nd-candidate gene(s): CAI= 0.6504,B479_12795;B479_20555;B479_21415;B479_22730;B479_22735;ppuh,Pseudomonas putida HB3267 acetolactate synthase I/II/III large subunit ;acetolactate synthase II small subunit;acetolactate synthase I/III small subunit 3rd-candidate gene(s): CAI= 0.642,T1E_0251;T1E_2130;T1E_2131;T1E_4866;T1E_5412,ppx,Pseudomonas putida DOT-T1E acetolactate synthase I/II/III large subunit ;acetolactate synthase II small subunit;acetolactate synthase I/III small subunit</p>

FIG. 3.7 Heterologous pathway for the production of (R,R)-2,3-butanediol shown as html-format

3.3.5 Examples of new heterologous pathways for the production of nonnative metabolites in the specific host

In previous part, the evaluation of the ArtPathDesign system has been done, thus, this part will demonstrate the advantage of this system that could identify a new heterologous pathway of nonnative metabolites in the target host cell. For example, muconic acid (C02480) previously reported to produce in *E. coli* by adding 3 heterologous reactions by enzymes 3-dehydroshikimate dehydratase, protocatechuic acid decarboxylase, and catechol 1,2-dioxygenase encoded by heterologous genes *aroZ*, *aroY* and *catA*, respectively (Niu et al., 2002). The new heterologous pathway was suggested by ArtPathDesign. This pathway contained 2 heterologous reactions by enzymes anthranilate 1,2-dioxygenase (EC number: 1.14.12.1) and catechol 1,2-dioxygenase (EC number: 1.13.11.1) encoded by heterologous genes *antABC* and *catA*, respectively. In addition, the new heterologous pathway of muconic acid was just reported by Sun et al., 2013 and it is identical with the new pathway suggested by ArtPathDesign. Sun and colleagues successfully produced muconic acid in *E. coli* by introducing the new pathway containing *antABC* and *catA* from *Pseudomonas aeruginosa* and *Pseudomonas putida* KT2440 (Sun et al., 2013). As the result, it is confirmed that ArtPathDesign successfully identified the new heterologous pathway for muconic acid and validated by the recently reported experiment (Sun et al., 2013).

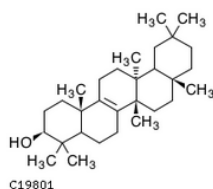
Another example is Isomultiflorenol production by using *E. coli* as the host (Fig. 3.8). Isomultiflorenol (Hayashi et al., 2001) is a triterpene showed similar structure with α -Amyrin and β -Amyrin (Fig. 3.9). In addition, α -Amyrin and β -

Amyrin (Liliana et al., 2012) have various pharmacological activities *in vitro* and *in vivo* conditions against various health-related conditions, including conditions such as anti-inflammation, antimicrobial, antifungal and antiviral infections. By using the ArtPathDesign system, it was demonstrated that 3 heterologous reactions are required to produce isomultiflorenol from trans,trans-Farnesyl diphosphate and 3 heterologous genes encoding 3 enzymes have activities of EC 2.5.1.21, EC 1.14.13.132, and EC 5.4.99.36. The candidate heterologous genes and open reading frame (ORF) ID are also suggested.

One more typical example is heterologous pathway for vanillin production in *E. coli* (Fig. 3.10). Vanillin is recognized as one of the most widely used flavoring agents in the world (Kaur and Chakraborty, 2013) which is extracted from the orchid *Vanilla planifolia*, *Vanilla tahitiensis*, and *Vanilla pompona*. In addition, pure vanillin used in food and beverage industry and also as a bio-preservative agent since its antimicrobial and antioxidant properties. According to vanillin properties, previous studies had design the heterologous pathway to incorporate into well-characterized hosts, *E. coli*, *S. cerevisiae*, etc. (Kaur and Chakraborty, 2013). By using the ArtPathDesign system, the new heterologous pathway and heterologous genes for *E. coli* as the host, which has no report today, are suggested. This heterologous pathway contains 4 heterologous reactions and 4 heterologous enzymes that have enzyme activities according to the following EC numbers: EC 1.14.13.33, EC 1.2.1.46, EC 1.14.13.82, and EC 1.2.1.67. The information of heterologous genes is shown below. Top substrates of this heterologous pathway are formate (C00058) and 4-hydroxybenzoate (C00156) which are able to be produced from the central metabolic pathway of *E. coli*.



Heterologous Pathway for C19801 in *E. coli* (Glucose as a carbon source)



Page created by: Sunisa Chatsurachai

Date: 2013-06-07

Target compound ID	C19801
Target compound Name	Isomultiflorenol
List of Heterologous reaction(s)	R06507 R04061 R14915
Detail of heterologous reaction	<p>R06507 (S)-squalene-2,3-epoxide <=> isomultiflorenol 1 C01054 <=> 1 C19801</p> <p>Enzyme information: 5.4.99.36 Gene symbol: NONE from database: EMBL 1st-candidate gene(s): CAI= 0.277,BAB68529,NONE,Luffa aegyptiaca (smooth loofah) isomultiflorenol synthase 2nd-candidate gene(s): CAI= NONE 3rd-candidate gene(s): CAI= NONE</p>
Detail of heterologous reaction	<p>R04061 squalene + NADPH + H⁺ + O₂ <=> (3S)-2,3-epoxy-2,3-dihydrosqualene + NADP⁺ + H₂O 1 C00751 + 1 C00005 + 1 C00080 + 1 C00007 <=> 1 C01054 + 1 C00006 + 1 C00001</p> <p>Enzyme information: 1.14.13.132 Gene symbol: SQLE,ERG1 from database: KEGG 1st-candidate gene(s): CAI= 0.47,LBRM_13_1480,lbz,Leishmania braziliensis squalene monooxygenase 2nd-candidate gene(s): CAI= 0.458,LMJF_13_1620,lma,Leishmania major squalene monooxygenase 3rd-candidate gene(s): CAI= 0.454,LDBPK_131360,ldo,Leishmania donovani squalene monooxygenase</p>
Detail of heterologous reaction	<p>R14915 2 trans,trans-Farnesyl diphosphate + NADPH + H⁺ <=> Squalene + 2 Diphosphate + NADP⁺ 2 C00448 + 1 C00005 + 1 C00080 <=> 1 C00751 + 2 C00013 + 1 C00006</p> <p>Enzyme information: 2.5.1.21 Gene symbol: FDFT1 from database: KEGG 1st-candidate gene(s): CAI= 0.514,LMJF_31_2940,lma,Leishmania major farnesyl-diphosphate farnesyltransferase 2nd-candidate gene(s): CAI= 0.502,LMXM_30_2940,lmi,Leishmania mexicana farnesyl-diphosphate farnesyltransferase 3rd-candidate gene(s): CAI= 0.501,Bcen_4104,bcn,Burkholderia cenocepacia AU1054 farnesyl-diphosphate farnesyltransferase</p>

FIG. 3.8 Heterologous pathway for Isomultiflorenol (C19801) production in *E. coli*

was produced via trans,trans-Farnesyl diphosphate (C00448).

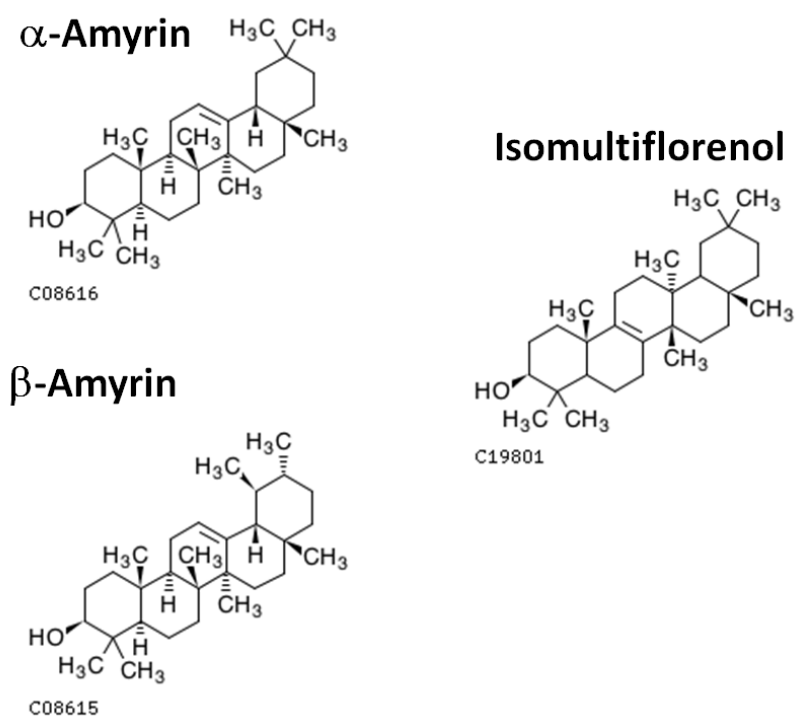
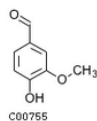


FIG. 3.9 Compound structures between α -Amyrin, β -Amyrin and Isomultiflorenol
(compound structures retrieved from KEGG database)

Heterologous Pathway for C00755 in *E. coli* (Glucose as a carbon source)



Page created by: Sunisa Chatsurachai

Date: 2013-06-07

Target compound ID	C00755
Target compound Name	Vanillin Vanillaldehyde 4-Hydroxy-3-methoxy-benzaldehyde 4-Hydroxy-3-methoxybenzaldehyde
List of Heterologous reaction(s)	R13873 R09195 R00285 R01792
Detail of heterologous reaction	<p>R13873 4-Hydroxy-3-methoxy-benzaldehyde + NAD⁺ + H₂O <=> Vanillate + NADH + H⁺ 1 C00755 + 1 C00003 + 1 C00001 <=> 1 C06672 + 1 C00004 + 1 C00080</p> <p>Enzyme information: 1.2.1.67 Gene symbol: NONE from database: EMEL 1st-candidate gene(s): CAI= 0.591,ABM36339,NONE,Polaromonas naphthalenivorans CJ2 vanillin dehydrogenase 2nd-candidate gene(s): CAI= 0.555,ADJ62666,NONE,Herbaspirillum seropedicae SmR1 vanillin: NAD dehydrogenase oxidoreductase protein 3rd-candidate gene(s): CAI= 0.544,ABB10168,NONE,Burkholderia sp. 383 vanillin dehydrogenase</p>
Detail of heterologous reaction	<p>R09195 vanillate + O₂ + NADH <=> 3,4-dihydroxybenzoate + NAD⁺ + H₂O + formaldehyde 1 C06672 + 1 C00007 + 1 C00004 <=> 1 C00230 + 1 C00003 + 1 C00001 + 1 C00067</p> <p>Enzyme information: 1.14.13.82 Gene symbol: vanA;vanB from database: KEGG 1st-candidate gene(s): CAI= 0.602,MDS_4193,pmk,Pseudomonas mendocina NK-01 vanillate monooxygenase;vanillate monooxygenase 2nd-candidate gene(s): CAI= 0.588,DSC_09845,DSC_09850,psd,Pseudoxanthomonas spadix vanillate monooxygenase;vanillate monooxygenase 3rd-candidate gene(s): CAI= 0.578,PP_3736,PP_3737,ppu,Pseudomonas putida KT2440 vanillate monooxygenase;vanillate monooxygenase</p>
Detail of heterologous reaction	<p>R00285 Formaldehyde + NAD⁺ + H₂O <=> Formate + NADH + H⁺ 1 C00067 + 1 C00003 + 1 C00001 <=> 1 C00058 + 1 C00004 + 1 C00080</p> <p>Enzyme information: 1.2.1.46 Gene symbol: fdhA from database: KEGG 1st-candidate gene(s): CAI= 0.702,PF01_5202,pfo,Pseudomonas fluorescens Pf0-1 glutathione-independent formaldehyde dehydrogenase 2nd-candidate gene(s): CAI= 0.696,PSF113_5429,pfe,Pseudomonas fluorescens F113 glutathione-independent formaldehyde dehydrogenase 3rd-candidate gene(s): CAI= 0.695,PP_0328,ppu,Pseudomonas putida KT2440 glutathione-independent formaldehyde dehydrogenase</p>
Detail of heterologous reaction	<p>R01792 p-hydroxybenzoate + O₂ <=> protocatechuic acid + H₂O 1 C00156 + 1 C00007 <=> 1 C00230 + 1 C00001</p> <p>Enzyme information: 1.1.1.24 Gene symbol: qa-3;QUIB from database: KEGG 1st-candidate gene(s): CAI= 0.492,MGG_07781,mgx,Magnaporthe oryzae quinate dehydrogenase 2nd-candidate gene(s): CAI= 0.431,ANI_1_1306034,ang,Aspergillus niger quinate dehydrogenase 3rd-candidate gene(s): CAI= 0.43,ACLA_024050,act,Aspergillus clavatus quinate dehydrogenase</p>

FIG. 3.10 Heterologous pathway for vanillin production in *E. coli*

3.4 Summary

An improved computational or *in silico* platform, named as ArtPathDesign, was developed to screen heterologous pathways suitable for the production of nonnative metabolites particular for host cell factories such as *E. coli*, *S. cerevisiae*, and *B. subtilis*. Owing to the presence of a vast number of candidate genes encoding similar enzymes and problem of low/no heterologous enzyme expression, the selection of suitable heterologous genes particular for the target host become difficult. Therefore, the applicability of the previously designed platform in chapter 2 was extended to select suitable candidate heterologous genes encoding enzymes. CAI score was used to screen the target genes whose introduction to host microorganisms would enable efficient target metabolite production. This ArtPathDesign system was improved to overcome the problems about host-specific heterologous pathways, host-specific heterologous genes, and enables to provide a catalog of nonnative metabolites in the specific host. Besides, the ArtPathDesign system could suggest the new heterologous pathways, for example Isomultiflorenol, vanillin, and so on. Furthermore, optimized nucleotide sequences of heterologous enzymes consisting of only the most preferred codons of hosts were calculated and these optimized heterologous genes may improve the production of metabolites if all optimized gene sequences applied for the corresponding heterologous pathway. It is expected that the *in silico* platform, ArtPathDesign, is proved as a valuable tool by providing essential information for improving cell factories. This aids researchers in developing strategies for strain improvement and facilitates the rational design of

metabolic pathways for the production of value-added chemicals by host microorganisms.

Chapter 4

4.1 General conclusion and discussion

Nowadays, the demands of fuels and chemical feedstocks are largely increased, while petroleum resources are limited and unsustainable. In addition, petroleum-based process for energy and industrial chemicals shows negative impacts on environment. The alternative route to produce fuels and valuable chemicals using microorganisms becomes a striking way. However, some microorganisms are not easy to cultivate and produce high level of target products, and lack of metabolic information and genetic manipulation tools.

Metabolic engineering is one of the most widely used techniques to modify and/or integrate heterologous pathways into the well-developed hosts, such as *E. coli*, *S. cerevisiae*, etc. for producing and/or improving target compounds. It is difficult for scientists to search for the feasible heterologous to produce target metabolites since the following reasons. There are a huge number of metabolic reactions available from databases and literature, and the complexity of host's metabolic network. To search for heterologous pathways connecting to the host's metabolism is tough and time-consuming tasks, and it is difficult to handle by human.

Accordingly, the computational or *in silico* system is required to solve those problems. In previous studies, several pathway design approaches (Cho et al., 2010; Dogrusoz et al., 2009; Finley et al., 2009; Flórez et al., 2011; Handorf et al., 2005; Li et

al., 2004; McShan et al., 2003; Moriya et al., 2010; Pey et al., 2011; Pharkya et al., 2004; Rodrigo et al., 2008; Yousofshahi et al., 2011) for the production of target metabolites have been developed (comparison of among methods are summarized in Table 1.2 in chapter 1). Nevertheless, there is no consensus or general approach to select heterologous pathways and genes especially for target host cells.

Therefore, the aim of this thesis was to develop the *in silico* method, named as the ArtPathDesign (Artificial heterologous Pathway Design), for screening host-specific heterologous pathways, host-specific heterologous genes and providing a catalog of nonnative metabolites of each host. These key features are important for producing/improving target compound in a specific host cell and still not yet be developed by the earlier reports.

In chapter 2, in order to produce target nonnative metabolites in the well-characterized hosts, *E. coli*, *S. cerevisiae*, and *C. glutamicum*, the algorithm to search for heterologous pathways was developed. With this algorithm, all possible nonnative metabolites that are able to connect to the 3 hosts were screened as well as information of heterologous reactions. Still, it was found that numerous orthologous genes encoding enzymes with similar function are available. Thus, a minimum of K_m value was applied with the expectation to have the highest affinity among orthologous genes to select candidate heterologous genes corresponding to heterologous pathway for the production of nonnative metabolites. Combining with the K_m selection score, the identical heterologous reactions of nonnative metabolites agreed well with widely used in metabolic engineering of industrial products were successfully screened (Table 2.2 and 2.3). The examples of those

nonnative metabolites are isoprene, α -farnesene, poly- β -hydroxybutyrate (PHB), and cadaverine. Furthermore, by comparing the number of producible nonnative metabolites among host microorganisms, it was found that yeast (*S. cerevisiae*) cell has several compartments presenting barriers to metabolite transport. For instance, mitochondrial/cytoplasmic interfaces prohibit the production of certain target metabolites when sugar is used as a carbon source. In addition, the genome-scale metabolic model used in this thesis does not account for transportation capabilities between compartments, which are currently unclear for many metabolites, and which might influence the production capacities of target metabolites in the real cell systems. By using the screening algorithm and K_m as the selection score, times and costs for searching suitable heterologous pathways agreed well with previous reports were reduced when comparing with the possible heterologous pathways generated without any background information. In addition, this computational method is applicable for any genome-scale metabolic models.

Although K_m value was used as the score for selection of candidate heterologous genes among numerous orthologous genes, the K_m information is limited since it required *in vitro* experiments to observe kinetic of enzymes and activities itself depending on substrate concentration. However, the expression level of a particular heterologous enzyme is crucial for the activation of a specific heterologous pathway to yield a higher productivity of the desired target metabolite.

In chapter 3, the improvement of the *in silico* platform was developed to facilitate the screening of host-specific pathways and genes by applying a new selection score. Firstly, the in-house database was updated from version 1.0 to 2.0 by

using the newest metabolic reaction database called BKM-react (Lang et al., 2011). The alternative score namely Codon Adaptation Index (CAI) was applied to be used as a selection score for finding candidate heterologous genes. The CAI score of a gene is calculated based on the preference codons found in the highly expressed genes of the host. It is expected that a higher CAI score resulting in a high expression level of the enzyme which could increase the production of target metabolite. With the CAI score as the selection score, the alternative heterologous genes were screened that could improve the productivity of nonnative metabolites such as 1, 3-propanediol in *S. cerevisiae* as the host. The heterologous pathway for (R, R)-2, 3 -butanediol in *E. coli* was identified and alternative heterologous genes were suggested. This *in silico* system, named as ArtPathDesign, is successfully identified the new heterologous pathways and genes of useful compounds such as cis,cis muconic acid, vanillin, Isomultiflorenol, etc. and could open the new biotransformation route to produce nonnative metabolites in the target host such as *E. coli*, *S. cerevisiae*, *B. subtilis*, etc. In case of cis,cis muconic acid, Sun and colleagues (Sun et al., 2013) was just reported the novel heterologous pathway which is the identical heterologous pathway suggested by the *in silico* system developed in this thesis. This example shows the proof of the algorithm successfully identified the new heterologous pathway that can be used to produce useful chemical in *E. coli* cell.

In conclusion, the computational platform, ArtPathDesign, was developed as a metabolic engineering tool in order to provide strategies for producing target metabolites in specific host cell factories, *E. coli*, *S. cerevisiae*, *C. glutamicum*, and *B. subtilis*. The ArtPathDesign platform is applicable for any genome-scale metabolic models. The strategy of current ArtPathDesign is shown in Fig. 4.1. Host's metabolic

networks and metabolic reactions from available databases were parsed into an in-house database, which will be used as the input data for generating all possible heterologous pathways. To select and rank heterologous genes for those possible heterologous pathways, the Codon Adaptation Index (CAI) was calculated for all orthologous genes retrieved from KEGG and EMBL databases and the optimized gene sequences were created by replacing original codons with favorable codons from the host's highly expressed genes without changing amino acid. Then, the rank of candidate heterologous genes and optimized gene sequences were included into the in-house database. Once all possible heterologous pathways were generated for the host cell, flux balance analysis (FBA) simulation was performed by including constraints that are glucose as a carbon source and mass balance. FBA was applied to observe capacity of the nonnative metabolites after introduction of heterologous pathway to the host metabolism under specific constraints. The heterologous pathway of the nonnative metabolite, which can produce more than 1% mol-product/mol-glucose, was selected and the heterologous genes corresponding to those heterologous pathways were ranked by using CAI score. Finally, list of nonnative metabolites, heterologous pathways, and genes was obtained to provide necessary information to users for conducting trial experiments. However, this platform was not included such factors thermodynamics of heterologous reactions, toxicity of products on the host cell, those features will be developed in the future as expansion features for suggesting more rational heterologous pathways.

The ArtPathDesign platform successfully identified host-specific heterologous pathways, host-specific heterologous genes, and provided the catalog of all nonnative metabolites are able to be produced by each host. Also, the new

heterologous pathways and genes were also suggested from this platform which would be a new route to engineer the host strain for target productions. Finally, the optimized gene sequences, which substituted original coding sequences with the preferable codons from highly expressed genes of each host, were also included. As mentioned in chapter 3, the cost for the artificial gene synthesis is gradually decreased. It is possible to improve the productivity of target metabolites by including optimized heterologous genes into the target host cell.

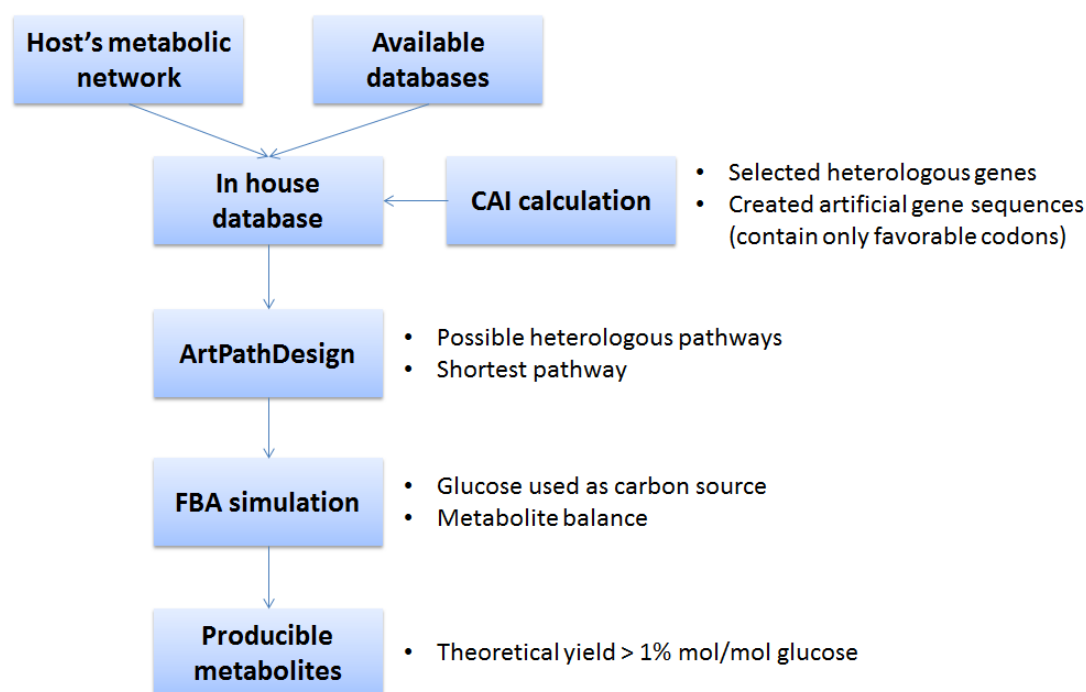


FIG. 4.1 Overall strategy of heterologous pathway identification and selection

(current version of ArtPathDesign system)

4.2 Future perspective

The current version of ArtPathDesign generates all possible heterologous pathways by assuming all known metabolic reactions to be reversible. However, there are a few class of reactions such as carboxylation/decarboxylation reaction, ATP consuming reactions (controlled by kinases), which are thermodynamically irreversible. In order to identify a more rational heterologous pathway, thermodynamics of heterologous reactions should be included to judge and select heterologous pathway for the production of nonnative metabolites. However, to manually incorporate all information on the reaction irreversibility is a hard task and it will be an expansion feature (shown in Fig. 4.2). Thermodynamics of metabolic reactions is calculated by using kinetic parameters retrieving from experiments, which are limitation. Several computational methods have been developed to estimate Gibbs free energy for biochemical reactions(Jankowski et al., 2008; Noor et al., 2012; Rother et al., 2010; Sabatini et al., 2012). One of these methods called group contribution is widely used. This group contribution method is demonstrated to be capable of estimating Gibbs free energy for the majority of the biochemical compounds and reactions found in *E. coli*'s metabolic network (Jankowski et al., 2008). Therefore, the reversibility of heterologous reactions will be calculated using this group contribution method and can be used to select more reasonable heterologous pathways. The thermodynamics feature should further develop for integrating with the current version of ArtPathDesign, and will help to improve the performance for selection of heterologous pathways.

The current version of ArtPathDesign platform was developed based on metabolic reactions data from available databases such as KEGG, BRENDA, BKM-react and so on. However, the input data for ArtPathDesign is depending on metabolic reactions from databases. Generating new metabolic reactions (non-existent in any databases) is a desire feature of heterologous pathway design. Therefore, to complete the ArtPathDesign system it requires a strategy to find new metabolic reactions, including information of enzymes/proteins and genes. Thus, another expansion feature is to create new heterologous reactions (Fig. 4.2 shown in red dashed box).

It was reported that several enzymes in *E. coli* (about 37% of enzymatic enzymes) that found to be generalist enzymes that promiscuously catalyze reactions on a variety of substrates (Nam et al., 2012). The idea of generalist enzymes leads to the possibility to create the novel metabolic reaction since these enzymes can bind to multi-substrates. Therefore, the design of novel metabolic reactions based on substrate similarity of generalist enzymes will be added to the ArtPathDesign as shown in Fig. 4.2 (red dashed box). Tanimoto or Jaccard coefficient is one of the most popular scores used to measure similarity between chemical structures represented by means of fingerprints (Willett et al., 1998). By using compound similarity score such as Tanimoto coefficient, the substrate-like compound will be screened. The possible new metabolic reaction will suggest from the substrate-like compound used as the alternative substrate in that metabolic reaction. The schematic of novel metabolic reaction design is shown in Fig. 4.3.

All metabolic enzymes and reactions from several databases such as KEGG, BRENDA, EMBL, ENZYMES, PDB, etc. will be compiled. Next, an enzyme that is able to catalyzed more than one reaction will be classified as generalists. Then, all substrate compounds of those generalist enzymes will be collected. Structures of those compounds can be retrieved from MetaCyc and KEGG databases. Consequently, Tanimoto coefficient will be calculated between the substrates of generalist enzymes and other compounds from available databases. If the similarity scores of such compounds can pass the threshold of the substrate-like compound, the compound will be used as the target for next step. After that, molecular docking technique will be used for observing enzyme-substrate affinity and a binding score will be calculated. Several docking tools (Yuriev and Ramsland, 2013) are available and successfully used for screening drug targets based on free energy binding and inhibition constant. In order to do enzyme-compound docking, target enzyme structures can be downloaded from PDB database. Finally, enzyme-binding scores between the substrate-like compound and target enzyme will be calculate by using tools such as AutoDock (Morris et al., 2009), GOLD (Jones et al., 1997), Glide (Friesner et al., 2004; Halgren et al., 2004), etc. However, the most important parameters are thresholds of substrate-like compounds and enzyme-compound binding scores; it required a computational system to repeat the process until finding a well-represented score which fits to some experimental data. Finally, new heterologous reactions will be identified and will be integrated to the ArtPathDesign system. Thus, to further complete ArtPathDesign system, the thermodynamics of heterologous reactions and the computational methods to design the new heterologous reaction(s) features will integrate to the current version of

ArtPathDesign. The complete version of ArtPathDesign would be the useful tool that enables the scientists for improvement of target metabolites in their target hosts by providing alternative metabolic engineering strategies.

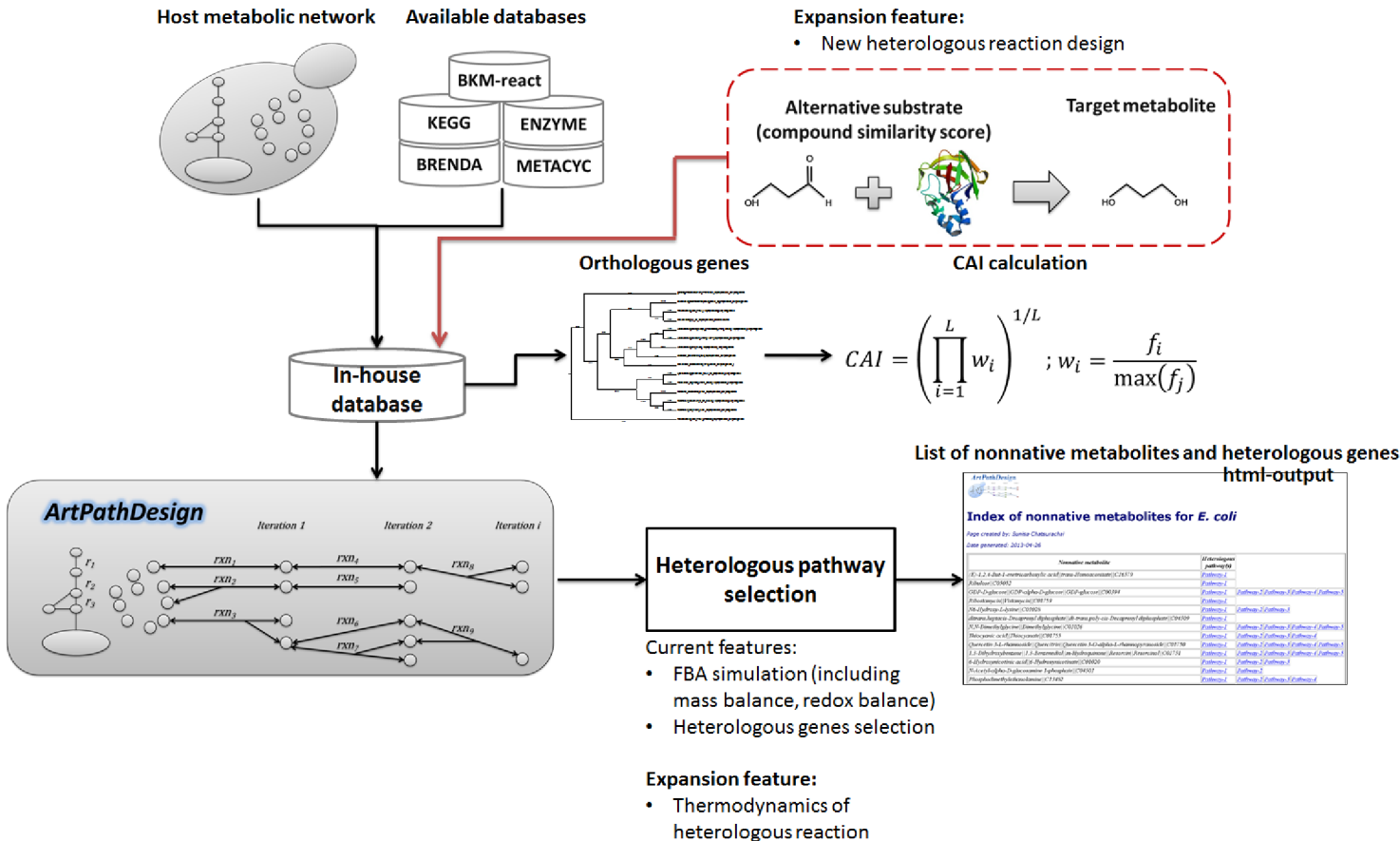


FIG. 4.2 The schematic of the complete ArtPathDesign integrated with the expansion features.

Thermodynamics of heterologous reactions will be expanded to improve the current version for selection of heterologous pathways. Red dashed box represents another expansion feature will be developed for creation of new heterologous reactions based on compound similarity and enzyme-substrate binding scores.

(An image of protein structure was retrieved from http://www.rcsb.org/pdb/images/2yow_bio_r_500.jpg)

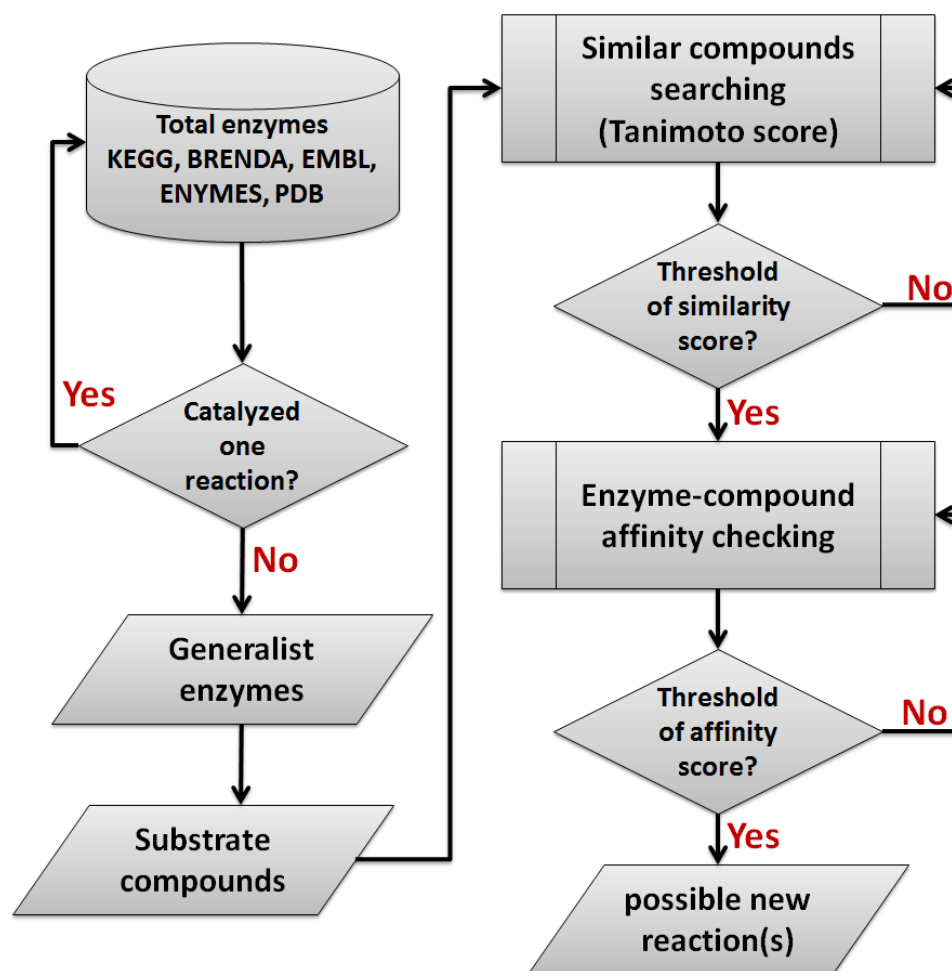


FIG. 4.3 The schematic of strategy to design new heterologous reactions

References

- Altaras, N. E., and Cameron, D. C. (1999).** Metabolic engineering of a 1,2-propanediol pathway in *Escherichia coli*. *Appl Environ Microbiol* **65**(3), 1180–1185.
- Angermayr, S. A., Paszota, M., and Hellingwerf, K. J. (2012).** Engineering a cyanobacterial cell factory for production of lactic acid. *Appl Environ Microbiol* **78**(19), 7098–7106.
- Bairoch, A. (2000).** The ENZYME database in 2000. *Nucleic Acids Res* **28**(1), 304–305.
- Becker, J., and Wittmann, C. (2012).** Bio-based production of chemicals, materials and fuels -*Corynebacterium glutamicum* as versatile cell factory. *Curr Opin Biotechnol* **23**(4), 631–640.
- Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Sayers, E. W. (2009).** GenBank. *Nucleic Acids Res* **37**(Database issue), D26–31.
- Biswas, R., Yamaoka, M., Nakayama, H., Kondo, T., Yoshida, K., Bisaria, V. S., and Kondo, A. (2012).** Enhanced production of 2,3-butanediol by engineered *Bacillus subtilis*. *Appl Microbiol Biotechnol* **94**(3), 651–658.
- Blattner, F. R., Plunkett, G., Bloch, C. A., Perna, N. T., Burland, V., Riley, M., et al. (1997).** The complete genome sequence of *Escherichia coli* K-12. *Science* **277**(5331), 1453–1462.
- Botzman, M., and Margalit, H. (2011).** Variation in global codon usage bias among prokaryotic organisms is associated with their lifestyles. *Genome Biol* **12**(10), R109.
- Burgard, A. P., Pharkya, P., and Maranas, C. D. (2003).** Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnol Bioeng* **84**(6), 647–657.
- Cameron, D. C., Altaras, N. E., Hoffman, M. L., and Shaw, A. J. (1998).** Metabolic engineering of propanediol pathways. *Biotechnol Prog* **14**(1), 116–125.
- Caspi, R., Foerster, H., Fulcher, C. A., Kaipa, P., Krummenacker, M., Latendresse, M., et al. (2008).** The MetaCyc Database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Res* **36**(Database issue), D623–31.

- Chang, A., Scheer, M., Grote, A., Schomburg, I., and Schomburg, D. (2009).** BRENDA, AMENDA and FRENDA the enzyme information system: new content and tools in 2009. *Nucleic Acids Res* **37**(Database issue), D588–92.
- Chatsurachai, S., Furusawa, C., and Shimizu, H. (2012).** An *in silico* platform for the design of heterologous pathways in nonnative metabolite production. *BMC Bioinformatics* **13**, 93.
- Chatsurachai, S., Furusawa, C., and Shimizu, H. (2013).** ArtPathDesign - Rational heterologous pathway design system for the production of nonnative metabolites. *J Biosci Bioeng.*(in press)
- Cho, A., Yun, H., Park, J. H., Lee, S. Y., and Park, S. (2010).** Prediction of novel synthetic pathways for the production of desired chemicals. *BMC Syst Biol* **4**, 35.
- Choi, H. S., Lee, S. Y., Kim, T. Y., and Woo, H. M. (2010).** *In silico* identification of gene amplification targets for improvement of lycopene production. *Appl Environ Microbiol* **76**(10), 3097–3105.
- Christina, S. D. (2010).** *The Metabolic Pathway Engineering Handbook: Fundamentals* (1st ed.). LLC, USA: CRC Press, Taylor& Francis Group.
- Clomburg, J. M., and Gonzalez, R. (2011).** Metabolic engineering of *Escherichia coli* for the production of 1,2-propanediol from glycerol. *Biotechnol Bioeng* **108**(4), 867–879.
- Cornelius, S. P., Lee, J. S., and Motter, A. E. (2011).** Dispensability of *Escherichia coli*'s latent pathways. *Proc Natl Acad Sci U S A* **108**(8), 3124–3129.
- Curran, K. A., Leavitt, J. M., Karim, A. S., and Alper, H. S. (2013).** Metabolic engineering of muconic acid production in *Saccharomyces cerevisiae*. *Metab Eng* **15**, 55–66.
- Dai, Z., Chen, Y., Qi, W., Huang, L., Zhang, Y., Zhou, T., Yang, X., and Gao, G. (2013).** Codon optimization increases human kallistatin expression in *Escherichia coli*. *Prep Biochem Biotechnol* **43**(1), 123–136.
- Das, S., Roymondal, U., and Sahoo, S. (2009).** Analyzing gene expression from relative codon usage bias in Yeast genome: a statistical significance and biological relevance. *Gene* **443**(1-2), 121–131.
- Dogrusoz, U., Cetintas, A., Demir, E., and Babur, O. (2009).** Algorithms for effective querying of compound graph-based pathway databases. *BMC Bioinformatics* **10**, 376.
- Dugar, D., and Stephanopoulos, G. (2011).** Relative potential of biosynthetic pathways for biofuels and bio-based products. *Nat Biotechnol* **29**(12), 1074–1078.

- Edwards, J. S., Ibarra, R. U., and Palsson, B. O. (2001). *In silico* predictions of *Escherichia coli* metabolic capabilities are consistent with experimental data. *Nat Biotechnol* **19**(2), 125–130.
- Edwards, J. S., and Palsson, B. O. (2000a). The *Escherichia coli* MG1655 *in silico* metabolic genotype: its definition, characteristics, and capabilities. *Proc Natl Acad Sci U S A* **97**(10), 5528–5533.
- Edwards, J. S., and Palsson, B. O. (2000b). Metabolic flux balance analysis and the *in silico* analysis of *Escherichia coli* K-12 gene deletions. *BMC Bioinformatics* **1**, 1.
- Feist, A. M., and Palsson, B. O. (2010). The biomass objective function. *Curr Opin Microbiol* **13**(3), 344–349.
- Festel, G., Detzel, C., and Mass, R. (2012). Industrial biotechnology — Markets and industry structure. *Journal of Commercial Biotechnology* **18**(1), 11–21.
- Finley, S. D., Broadbelt, L. J., and Hatzimanikatis, V. (2009). Computational framework for predictive biodegradation. *Biotechnol Bioeng* **104**(6), 1086–1097.
- Fleischmann, R. D., Adams, M. D., White, O., Clayton, R. A., Kirkness, E. F., Kerlavage, A. R., et al. (1995). Whole-genome random sequencing and assembly of *Haemophilus influenzae* Rd. *Science* **269**(5223), 496–512.
- Flórez, L. A., Gunka, K., Polanía, R., Tholen, S., and Stülke, J. (2011). SPABBATS: A pathway-discovery method based on Boolean satisfiability that facilitates the characterization of suppressor mutants. *BMC Syst Biol* **5**, 5.
- Fong, S. S., Marciniak, J. Y., and Palsson, B. Ø. (2003). Description and interpretation of adaptive evolution of *Escherichia coli* K-12 MG1655 by using a genome-scale *in silico* metabolic model. *J Bacteriol* **185**(21), 6400–6408.
- Friesner, R. A., Banks, J. L., Murphy, R. B., Halgren, T. A., Klicic, J. J., Mainz, D. T., et al. (2004). Glide: a new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy. *J Med Chem* **47**(7), 1739–1749.
- Futcher, B., Latter, G. I., Monardo, P., McLaughlin, C. S., and Garrels, J. I. (1999). A sampling of the yeast proteome. *Mol Cell Biol* **19**(11), 7357–7368.
- Gerdes, S. Y., Scholle, M. D., Campbell, J. W., Balázsi, G., Ravasz, E., Daugherty, M. D., et al. (2003). Experimental determination and system level analysis of essential genes in *Escherichia coli* MG1655. *J Bacteriol* **185**(19), 5673–5684.
- Goffeau, A., Barrell, B. G., Bussey, H., Davis, R. W., Dujon, B., Feldmann, H., et al. (1996). Life with 6000 genes. *Science* **274**(5287), 546, 563–567.

- Guadalupe Medina, V., Almering, M. J. H., Van Maris, A. J. A., and Pronk, J. T. (2010).** Elimination of glycerol production in anaerobic cultures of a *Saccharomyces cerevisiae* strain engineered to use acetic acid as an electron acceptor. *Appl Environ Microbiol* **76**(1), 190–195.
- Gupta, S. K., Bhattacharyya, T. K., and Ghosh, T. C. (2004).** Synonymous codon usage in *Lactococcus lactis*: mutational bias versus translational selection. *J Biomol Struct Dyn* **21**(4), 527–536.
- Gvritishvili, A. G., Leung, K. W., and Tombran-Tink, J. (2010).** Codon preference optimization increases heterologous PEDF expression. *PLoS One* **5**(11), e15056.
- Halgren, T. A., Murphy, R. B., Friesner, R. A., Beard, H. S., Frye, L. L., Pollard, W. T., and Banks, J. L. (2004).** Glide: a new approach for rapid, accurate docking and scoring. 2. Enrichment factors in database screening. *J Med Chem* **47**(7), 1750–1759.
- Handorf, T., Ebenhöf, O., and Heinrich, R. (2005).** Expanding metabolic networks: scopes of compounds, robustness, and evolution. *J Mol Evol* **61**(4), 498–512.
- Hayashi, H., Huang, P., Inoue, K., Hiraoka, N., Ikeshiro, Y., Yazaki, K., et al. (2001).** Molecular cloning and characterization of isomultiflorenol synthase, a new triterpene synthase from *Luffa cylindrica*, involved in biosynthesis of bryonolic acid. *Eur J Biochem* **268**(23), 6311–6317.
- Hoefel, T., Faust, G., Reinecke, L., Rudinger, N., and Weuster-Botz, D. (2012).** Comparative reaction engineering studies for succinic acid production from sucrose by metabolically engineered *Escherichia coli* in fed-batch-operated stirred tank bioreactors. *Biotech J* **7**(10), 1277–1287.
- Hong, K.-K., and Nielsen, J. (2012).** Metabolic engineering of *Saccharomyces cerevisiae*: a key cell factory platform for future biorefineries. *Cell Mol Life Sci* **69**(16), 2671–2690.
- Ingvarsson, P. K. (2008).** Molecular evolution of synonymous codon usage in *Populus*. *BMC Evol Biol* **8**, 307.
- Inui, M., Kawaguchi, H., Murakami, S., Vertès, A. A., and Yukawa, H. (2004).** Metabolic engineering of *Corynebacterium glutamicum* for fuel ethanol production under oxygen-deprivation conditions. *J Mol Microbiol Biotechnol* **8**(4), 243–254.
- Jankowski, M. D., Henry, C. S., Broadbelt, L. J., and Hatzimanikatis, V. (2008).** Group contribution method for thermodynamic analysis of complex metabolic networks. *Biophysical journal* **95**(3), 1487–1499.
- Jia, X., Li, S., Xie, S., and Wen, J. (2012).** Engineering a metabolic pathway for isobutanol biosynthesis in *Bacillus subtilis*. *Appl Biochem Biotechnol* **168**(1), 1–9.

- Jones, G., Willett, P., Glen, R. C., Leach, A. R., and Taylor, R. (1997). Development and validation of a genetic algorithm for flexible docking. *J Mol Biol* **267**(3), 727–748.
- Juminaga, D., Baidoo, E. E. K., Redding-Johanson, A. M., Batth, T. S., Burd, H., Mukhopadhyay, A., Petzold, C. J., and Keasling, J. D. (2012). Modular engineering of L-tyrosine production in *Escherichia coli*. *Appl Environ Microbiol* **78**(1), 89–98.
- Kalinowski, J., Bathe, B., Bartels, D., Bischoff, N., Bott, M., Burkovski, A., et al. (2003). The complete *Corynebacterium glutamicum* ATCC 13032 genome sequence and its impact on the production of L-aspartate-derived amino acids and vitamins. *J Biotechnol* **104**(1-3), 5–25.
- Kanehisa, M., Araki, M., Goto, S., Hattori, M., Hirakawa, M., Itoh, M., et al. (2008). KEGG for linking genomes to life and the environment. *Nucleic Acids Res* **36**(Database issue), D480–4.
- Kauffman, K. J., Prakash, P., and Edwards, J. S. (2003). Advances in flux balance analysis. *Curr Opin Biotechnol* **14**(5), 491–496.
- Kaur, B., and Chakraborty, D. (2013). Biotechnological and molecular approaches for vanillin production: a review. *Appl Biochem Biotechnol* **169**(4), 1353–1372.
- Kind, S., Jeong, W. K., Schröder, H., and Wittmann, C. (2010). Systems-wide metabolic pathway engineering in *Corynebacterium glutamicum* for bio-based production of diaminopentane. *Metab Eng* **12**(4), 341–351.
- Kind, S., and Wittmann, C. (2011). Bio-based production of the platform chemical 1,5-diaminopentane. *Appl Microbiol Biotechnol* **91**(5), 1287–1296.
- Kleber-Janke, T., and Becker, W. M. (2000). Use of modified BL21(DE3) *Escherichia coli* cells for high-level expression of recombinant peanut allergens affected by poor codon usage. *Protein Expr Purif* **19**(3), 419–424.
- Kuenz, A., Gallenmüller, Y., Willke, T., and Vorlop, K.-D. (2012). Microbial production of itaconic acid: developing a stable platform for high product concentrations. *Appl Microbiol Biotechnol* **96**(5), 1209–1216.
- Kulikova, T., Aldebert, P., Althorpe, N., Baker, W., Bates, K., Browne, P., et al. (2004). The EMBL Nucleotide Sequence Database. *Nucleic Acids Res* **32**(Database issue), D27–30.
- Kunst, F., Ogasawara, N., Moszer, I., Albertini, A. M., Alloni, G., Azevedo, V., et al. (1997). The complete genome sequence of the gram-positive bacterium *Bacillus subtilis*. *Nature* **390**(6657), 249–256.

- Lakey, D. L., Voladri, R. K., Edwards, K. M., Hager, C., Samten, B., Wallis, R. S., Barnes, P. F., and Kernodle, D. S. (2000).** Enhanced production of recombinant *Mycobacterium tuberculosis* antigens in *Escherichia coli* by replacement of low-usage codons. *Infect Immun* **68**(1), 233–238.
- Lan, E. I., and Liao, J. C. (2012).** ATP drives direct photosynthetic production of 1-butanol in cyanobacteria. *Proc Natl Acad Sci U S A* **109**(16), 6018–6023.
- Lang, M., Stelzer, M., and Schomburg, D. (2011).** BKM-react, an integrated biochemical reaction database. *BMC Biochem* **12**, 42.
- Lee, J. W., Kim, H. U., Choi, S., Yi, J., and Lee, S. Y. (2011).** Microbial production of building block chemicals and polymers. *Curr Opin Biotechnol* **22**(6), 758–767.
- Lee, M.-S., Hseu, Y.-C., Lai, G.-H., Chang, W.-T., Chen, H.-J., Huang, C.-H., et al. (2011).** High yield expression in a recombinant *E. coli* of a codon optimized chicken anemia virus capsid protein VP1 useful for vaccine development. *Microb Cell Fact* **10**, 56.
- Lee, S. K., Chou, H., Ham, T. S., Lee, T. S., and Keasling, J. D. (2008).** Metabolic engineering of microorganisms for biofuels production: from bugs to synthetic biology to fuels. *Curr Opin Biotechnol* **19**(6), 556–563.
- Lee, S., Kim, B., Oh, M., Kim, Y., and Lee, J. (2012).** Enhanced activity of meso-secondary alcohol dehydrogenase from *Klebsiella* species by codon optimization. *Bioprocess and biosystems engineering* **36**(7), 1005–1010.
- Lee, W., and Dasilva, N. A. (2006).** Application of sequential integration for metabolic engineering of 1,2-propanediol production in yeast. *Metab Eng* **8**(1), 58–65.
- Li, C., Henry, C. S., Jankowski, M. D., Ionita, J. A., Hatzimanikatis, V., and Broadbelt, L. J. (2004).** Computational discovery of biochemical routes to specialty chemicals. *Chem Eng Sci* **59**(22–23), 5051–5060.
- Li, W., Ng, I.-S., Fang, B., Yu, J., and Zhang, G. (2011).** Codon optimization of 1,3-propanediol oxidoreductase expression in *Escherichia coli* and enzymatic properties. *Electron J Biotechnol* **14**(4).
- Liliana, H. V., Javier, P., and Arturo, N. O. (2012).** The Pentacyclic Triterpenes alpha, beta-amyrins: A Review of Sources and Biological Activities. *Phytochemicals - A Global Perspective of Their Role in Nutrition and Health* (Vol. 426). InTech.
- Lindahl, A.-L., Olsson, M. E., Mercke, P., Tollbom, O., Schelin, J., Brodelius, M., and Brodelius, P. E. (2006).** Production of the artemisinin precursor amorpha-4,11-diene by engineered *Saccharomyces cerevisiae*. *Biotechnol Lett* **28**(8), 571–580.

- Litsanov, B., Kabus, A., Brocker, M., and Bott, M. (2012).** Efficient aerobic succinate production from glucose in minimal medium with *Corynebacterium glutamicum*. *Microbial biotechnology* **5**(1), 116–128.
- Liu, G., Wu, J., Yang, H., and Bao, Q. (2010).** Codon Usage Patterns in *Corynebacterium glutamicum*: Mutational Bias, Natural Selection and Amino Acid Conservation. *Comp Funct Genomics* **2010**, 343569.
- Liu, H., Xu, Y., Zheng, Z., and Liu, D. (2010).** 1,3-Propanediol and its copolymers: research, development and industrialization. *Biotech J* **5**(11), 1137–1148.
- Lv, X., Xu, H., and Yu, H. (2013).** Significantly enhanced production of isoprene by ordered coexpression of genes *dxs*, *dxr*, and *idi* in *Escherichia coli*. *Appl Microbiol Biotechnol* **97**(6), 2357–2365.
- Mahishi, L. H., Tripathi, G., and Rawal, S. K. (2003).** Poly(3-hydroxybutyrate) (PHB) synthesis by recombinant *Escherichia coli* harbouring *Streptomyces aureofaciens* PHB biosynthesis genes: effect of various carbon and nitrogen sources. *Microbiol Res* **158**(1), 19–27.
- Mani, I., Singh, V., Chaudhary, D. K., Somvanshi, P., and Negi, M. P. S. (2011).** Codon optimization of the major antigen encoding genes of diverse strains of influenza a virus. *Interdiscip Sci* **3**(1), 36–42.
- Martín-Galiano, A. J., Wells, J. M., and De la Campa, A. G. (2004).** Relationship between codon biased genes, microarray expression values and physiological characteristics of *Streptococcus pneumoniae*. *Microbiology* **150**(Pt 7), 2313–2325.
- Mazumdar, S., Blankschien, M. D., Clomburg, J. M., and Gonzalez, R. (2013).** Efficient synthesis of L-lactic acid from glycerol by metabolically engineered *Escherichia coli*. *Microb Cell Fact* **12**(1), 7.
- McKendry, P. (2002).** Energy production from biomass (Part 1): Overview of biomass. *Bioresource technology* **83**(1), 37–46.
- McShan, D. C., Rao, S., and Shah, I. (2003).** PathMiner: predicting metabolic pathways by heuristic search. *Bioinformatics* **19**(13), 1692–1698.
- Mo, M. L., Palsson, B. O., and Herrgård, M. J. (2009).** Connecting extracellular metabolomic measurements to intracellular flux states in yeast. *BMC Syst Biol* **3**, 37.
- Moon, T. S., Yoon, S.-H., Lanza, A. M., Roy-Mayhew, J. D., and Prather, K. L. J. (2009).** Production of glucaric acid from a synthetic pathway in recombinant *Escherichia coli*. *Appl Environ Microbiol* **75**(3), 589–595.

- Moriya, Y., Shigemizu, D., Hattori, M., Tokimatsu, T., Kotera, M., Goto, S., and Kanehisa, M. (2010).** PathPred: an enzyme-catalyzed metabolic pathway prediction server. *Nucleic Acids Res* **38**(Web Server issue), W138–43.
- Morris, G. M., Huey, R., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S., and Olson, A. J. (2009).** AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *J Comput Chem* **30**(16), 2785–2791.
- Nakamura, C. E., and Whited, G. M. (2003).** Metabolic engineering for the microbial production of 1,3-propanediol. *Curr Opin Biotechnol* **14**(5), 454–459.
- Nam, H., Lewis, N. E., Lerman, J. A., Lee, D.-H., Chang, R. L., Kim, D., and Palsson, B. O. (2012).** Network context and selection in the evolution to enzyme specificity. *Science* **337**(6098), 1101–1104.
- Nayak, K. C. (2012).** Comparative study on factors influencing the codon and amino acid usage in *Lactobacillus sakei* 23K and 13 other lactobacilli. *Mol Biol Rep* **39**(1), 535–545.
- NC-IUBMB. (1999).** IUPAC-IUBMB Joint Commission on Biochemical Nomenclature (JCBN) and Nomenclature Committee of IUBMB (NC-IUBMB), newsletter 1999. *Eur J Biochem* **264**(2), 607–609.
- Nielsen, D. R., Yoon, S.-H., Yuan, C. J., and Prather, K. L. J. (2010).** Metabolic engineering of acetoin and meso-2, 3-butanediol biosynthesis in *E. coli*. *Biotech J* **5**(3), 274–284.
- Niu, W., Draths, K. M., and Frost, J. W. (2002).** Benzene-free synthesis of adipic acid. *Biotechnol Prog* **18**(2), 201–211.
- Noor, E., Bar-Even, A., Flamholz, A., Lubling, Y., Davidi, D., and Milo, R. (2012).** An integrated open framework for thermodynamics of reactions that combines accuracy and coverage. *Bioinformatics* **28**(15), 2037–2044.
- Norkiene, M., and Gedvilaite, A. (2012).** Influence of codon bias on heterologous production of human papillomavirus type 16 major structural protein L1 in yeast. *ScientificWorldJournal* **2012**, 979218.
- Oh, Y.-K., Palsson, B. O., Park, S. M., Schilling, C. H., and Mahadevan, R. (2007).** Genome-scale reconstruction of metabolic network in *Bacillus subtilis* based on high-throughput phenotyping and gene essentiality data. *J Biol Chem* **282**(39), 28791–28799.
- Ohya, N., and Koyama, T. (2005).** *Biopolymers Online* (pp. 73–81). Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA.

- Olivares-Hernández, R., Bordel, S., and Nielsen, J. (2011).** Codon usage variability determines the correlation between proteome and transcriptome fold changes. *BMC Syst Biol* **5**, 33.
- Orth, J. D., Thiele, I., and Palsson, B. Ø. (2010).** What is flux balance analysis? *Nat Biotechnol* **28**(3), 245–248.
- Otero, J. M., Cimini, D., Patil, K. R., Poulsen, S. G., Olsson, L., and Nielsen, J. (2013).** Industrial Systems Biology of *Saccharomyces cerevisiae* Enables Novel Succinic Acid Cell Factory. *PLoS One* **8**(1), e54144.
- Pacheco, A., Talaia, G., Sá-Pessoa, J., Bessa, D., Gonçalves, M. J., Moreira, R., Paiva, S., Casal, M., and Queirós, O. (2012).** Lactic acid production in *Saccharomyces cerevisiae* is modulated by expression of the monocarboxylate transporters Jen1 and Ady2. *FEMS yeast research* **12**(3), 375–381.
- Pagani, I., Liolios, K., Jansson, J., Chen, I.-M. A., Smirnova, T., Nosrat, B., Markowitz, V. M., and Kyrpides, N. C. (2012).** The Genomes OnLine Database (GOLD) v.4: status of genomic and metagenomic projects and their associated metadata. *Nucleic Acids Res* **40**(Database issue), D571–9.
- Papini, M., Salazar, M., and Nielsen, J. (2010).** Systems biology of industrial microorganisms. *Adv Biochem Eng Biotechnol* **120**, 51–99.
- Park, J. H., Lee, K. H., Kim, T. Y., and Lee, S. Y. (2007).** Metabolic engineering of *Escherichia coli* for the production of L-valine based on transcriptome analysis and *in silico* gene knockout simulation. *Proc Natl Acad Sci U S A* **104**(19), 7797–7802.
- Pey, J., Prada, J., Beasley, J. E., and Planes, F. J. (2011).** Path finding methods accounting for stoichiometry in metabolic networks. *Genome Biol* **12**(5), R49.
- Pharkya, P., Burgard, A. P., and Maranas, C. D. (2004).** OptStrain: a computational framework for redesign of microbial production systems. *Genome Res* **14**(11), 2367–2376.
- Puigbò, P., Romeu, A., and Garcia-Vallvé, S. (2008).** HEG-DB: a database of predicted highly expressed genes in prokaryotic complete genomes under translational selection. *Nucleic Acids Res* **36**(Database issue), D524–7.
- Qian, W., Yang, J.-R., Pearson, N. M., Maclean, C., and Zhang, J. (2012).** Balanced codon usage optimizes eukaryotic translational efficiency. *PLoS Genet* **8**(3), e1002603.
- Qian, Z.-G., Xia, X.-X., and Lee, S. Y. (2011).** Metabolic engineering of *Escherichia coli* for the production of cadaverine: a five carbon diamine. *Biotechnol Bioeng* **108**(1), 93–103.

- Raman, K., and Chandra, N. (2009).** Flux balance analysis of biological systems: applications and challenges. *Briefings in bioinformatics* **10**(4), 435–449.
- Raman, K., Rajagopalan, P., and Chandra, N. (2006).** Principles and Practices of Pathway Modelling. *Current Bioinformatics* **1**(2), 147–160.
- Rao, Z., Ma, Z., Shen, W., Fang, H., Zhuge, J., and Wang, X. (2008).** Engineered *Saccharomyces cerevisiae* that produces 1,3-propanediol from D-glucose. *J Appl Microbiol* **105**(6), 1768–1776.
- Rathnasingh, C., Raj, S. M., Lee, Y., Catherine, C., Ashok, S., and Park, S. (2012).** Production of 3-hydroxypropionic acid via malonyl-CoA pathway using recombinant *Escherichia coli* strains. *J Biotechnol* **157**(4), 633–640.
- Redemann, S., Schloissnig, S., Ernst, S., Pozniakowsky, A., Ayloo, S., Hyman, A. A., and Bringmann, H. (2011).** Codon adaptation-based control of protein expression in *C. elegans*. *Nat Methods* **8**(3), 250–252.
- Reed, J. L., Vo, T. D., Schilling, C. H., and Palsson, B. O. (2003).** An expanded genome-scale model of *Escherichia coli* K-12 (iJR904 GSM/GPR). *Genome Biol* **4**(9), R54.
- Rodrigo, G., Carrera, J., Prather, K. J., and Jaramillo, A. (2008).** DESHARKY: automatic design of metabolic pathways for optimal cell growth. *Bioinformatics* **24**(21), 2554–2556.
- Rother, K., Hoffmann, S., Bulik, S., Hoppe, A., Gasteiger, J., and Holzhütter, H.-G. (2010).** IGERS: inferring Gibbs energy changes of biochemical reactions from reaction similarities. *Biophysical journal* **98**(11), 2478–2486.
- Sabatini, A., Vacca, A., and Iotti, S. (2012).** Balanced biochemical reactions: a new approach to unify chemical and biochemical thermodynamics. *PLoS One* **7**(1), e29529.
- Schneider, J., and Wendisch, V. F. (2011).** Biotechnological production of polyamines by bacteria: recent achievements and future perspectives. *Appl Microbiol Biotechnol* **91**(1), 17–30.
- Schuetz, R., Kuepfer, L., and Sauer, U. (2007).** Systematic evaluation of objective functions for predicting intracellular fluxes in *Escherichia coli*. *Mol Syst Biol* **3**, 119.
- Sharp, P. M., and Li, W. H. (1987).** The codon Adaptation Index--a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res* **15**(3), 1281–1295.

- Sharp, P. M., Tuohy, T. M., and Mosurski, K. R. (1986).** Codon usage in yeast: cluster analysis clearly differentiates highly and lowly expressed genes. *Nucleic Acids Res* **14**(13), 5125–5143.
- Shen, C. R., Lan, E. I., Dekishima, Y., Baez, A., Cho, K. M., and Liao, J. C. (2011).** Driving forces enable high-titer anaerobic 1-butanol synthesis in *Escherichia coli*. *Appl Environ Microbiol* **77**(9), 2905–2915.
- Shinfuku, Y., Sorpitiporn, N., Sono, M., Furusawa, C., Hirasawa, T., and Shimizu, H. (2009).** Development and experimental verification of a genome-scale metabolic model for *Corynebacterium glutamicum*. *Microb Cell Fact* **8**, 43.
- Soyer, O. S., and Pfeiffer, T. (2010).** Evolution under fluctuating environments explains observed robustness in metabolic networks. *PLoS Comput Biol* **6**(8).
- Sun, X., Lin, Y., Huang, Q., Yuan, Q., and Yan, Y. (2013).** A Novel Muconic Acid Biosynthetic Approach by Shunting Tryptophan Biosynthesis via Anthranilate. *Appl Environ Microbiol* **79**(13), 4024–4030.
- Tao, P., Dai, L., Luo, M., Tang, F., Tien, P., and Pan, Z. (2009).** Analysis of synonymous codon usage in classical swine fever virus. *Virus Genes* **38**(1), 104–112.
- Varma, A., and Palsson, B. O. (1994).** Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type *Escherichia coli* W3110. *Appl Environ Microbiol* **60**(10), 3724–3731.
- Vennestrøm, P. N. R., Osmundsen, C. M., Christensen, C. H., and Taarning, E. (2011).** Beyond petrochemicals: the renewable chemicals industry. *Angewandte Chemie (International ed in English)* **50**(45), 10502–10509.
- Wallaart, T. E., Bouwmeester, H. J., Hille, J., Poppinga, L., and Maijers, N. C. (2001).** Amorpha-4,11-diene synthase: cloning and functional expression of a key enzyme in the biosynthetic pathway of the novel antimalarial drug artemisinin. *Planta* **212**(3), 460–465.
- Wang, B.-W., Shi, A.-Q., Tu, R., Zhang, X.-L., Wang, Q.-H., and Bai, F.-W. (2012).** Branched-chain higher alcohols. *Adv Biochem Eng Biotechnol* **128**, 101–118.
- Wang, C., Yoon, S.-H., Jang, H.-J., Chung, Y.-R., Kim, J.-Y., Choi, E.-S., and Kim, S.-W. (2011).** Metabolic engineering of *Escherichia coli* for α -farnesene production. *Metab Eng* **13**(6), 648–655.
- Wang, M., Weiss, M., Simonovic, M., Haertinger, G., Schrimpf, S. P., Hengartner, M. O., and Von Mering, C. (2012).** PaxDb, a database of protein abundance averages across all three domains of life. *Mol Cell Proteomics* **11**(8), 492–500.

- Wang, Q., Mei, C., Zhen, H., and Zhu, J. (2012).** Codon preference optimization increases prokaryotic cystatin C expression. *J Biomed Biotechnol* **2012**, 732017.
- Washburn, M. P., Wolters, D., and Yates, J. R. (2001).** Large-scale analysis of the yeast proteome by multidimensional protein identification technology. *Nat Biotechnol* **19**(3), 242–247.
- Willett, P., Barnard, J. M., and Downs, G. M. (1998).** Chemical Similarity Searching. *J Chem Inf Model* **38**(6), 983–996.
- Woodruff, L. B. A., May, B. L., Warner, J. R., and Gill, R. T. (2013).** Towards a metabolic engineering strain “commons”: An *Escherichia coli* platform strain for ethanol production. *Biotechnol Bioeng* **110**(5), 1520–1526.
- Yan, Y., Lee, C.-C., and Liao, J. C. (2009).** Enantioselective synthesis of pure (R,R)-2,3-butanediol in *Escherichia coli* with stereospecific secondary alcohol dehydrogenases. *Org Biomol Chem* **7**(19), 3914–3917.
- Yim, H., Haselbeck, R., Niu, W., Pujol-Baxley, C., Burgard, A., Boldt, J., et al. (2011).** Metabolic engineering of *Escherichia coli* for direct production of 1,4-butanediol. *Nature chemical biology* **7**(7), 445–452.
- Yousofshahi, M., Lee, K., and Hassoun, S. (2011).** Probabilistic pathway construction. *Metab Eng* **13**(4), 435–444.
- Yuriev, E., and Ramsland, P. A. (2013).** Latest developments in molecular docking: 2010–2011 in review. *Journal of molecular recognition : JMR* **26**(5), 215–239.
- Zhang, X., Wang, X., Shanmugam, K. T., and Ingram, L. O. (2011).** L-malate production by metabolically engineered *Escherichia coli*. *Appl Environ Microbiol* **77**(2), 427–434.
- Zhao, Y., Yang, J., Qin, B., Li, Y., Sun, Y., Su, S., and Xian, M. (2011).** Biosynthesis of isoprene in *Escherichia coli* via methylerythritol phosphate (MEP) pathway. *Appl Microbiol Biotechnol* **90**(6), 1915–1922.
- Zweers, J. C., Barák, I., Becher, D., Driessen, A. J., Hecker, M., Kontinen, V. P., Saller, M. J., Vavrová, L., and Van Dijk, J. M. (2008).** Towards the development of *Bacillus subtilis* as a cell factory for membrane proteins and protein complexes. *Microb Cell Fact* **7**, 10.

Appendix A

Source code

A.1 Heterologous pathways detection by Python™

A.1.1 Main script “runningScript.py”

```
#####
### Runing script by using this file with IDLE
### Before run the script, you need to change the path of input file that
you want to obtain all possible connected metabolites
### kegg_metabolic_reaction2010mar.csv -> is the reference reactions from
KEGG ligand database
### host_metabolites.txt -> contains all metabolites in terms of KEGG ID if
available ( you can get this list fromn the supplementary of Genome-scale
metabolic model
### nonNativeMetabolites.txt -> contains all metabolites (retriving from
reference reactions that are not in the host metabolic network)
###
#####

import sys,pickle,re,os,csv,glob
from heterogeneousReactionDetection import *
from addInfoOutput import *
from createIndexFile import *

## current directory
cur_dir = os.getcwd()

### Three input files are neccessary for running the script #####
### Please see the format of input files in the input folder #####
bkm_rn_file =
os.path.join(cur_dir,"input/bkm_metabolic_reactions_2012.csv")## KEGG
reaction in tab delimited format
gemMet_file = os.path.join(cur_dir,"input/eco_nativeMetbolites.csv")## host
metabolites if no KEGG ID used the original ID
```

```
nonNative_file =
os.path.join(cur_dir,"input/eco_nonNativeMetabolites.csv")## nonnative
metabolites ( not found in host metabolic network)
#####

### list of metabolites from host metabolic model
gemMetList = [line.upper().rstrip() for line in open(gemMet_file)]

### list of nonnative metabolites (found in KEGG reference reactions, but
not found in host network)
nonNativeMetList = [line.upper().rstrip() for line in open(nonNative_file)]

##### Creating Dictionary of KEGG reactions #####
result = Reaction(bkm_rn_file)## class KEGG metabolic reaction with
reaction ID, left metabolites, right metabolite
bkm_rns={}
for item in result.getReaction():
    rec =
[item.rx_id,item.leftMet,item.rightMet,item.equation,item.orgEquation,item.
direction]
    bkm_rns[item.rx_id] = rec

#####

### Input the number of iteration that you want to observe
numOfiteration = raw_input("Please input total iterations (such as 10,20,
or 30) and then press Enter = ")

### To write pre-results that contain the possible reaction to be used
maxIteration =
writePreResult(numOfiteration,nonNativeMetList,bkm_rns,gemMetList)

### To write all heterologous reactions (equation in term of KEGG compound
ID [Cxxxxx]) for each nonnative metabolites in each iteration
writeAllHeterologousPathwayToFile(maxIteration,bkm_rns)
```

```
print "Finished searching !!!!!!! for the current host organism ....."
```

```
## To remove preResult file ( *.csv )  
removePreResultFile()
```

```
## To generate output files in html-format ####  
input_dir = os.path.join(cur_dir,"input")
```

```
bkmReactDict =  
getBKMreaction(os.path.join(input_dir,"bkm_metabolic_reactions_2012.csv"))  
keggDict = getKEGGcompound(os.path.join(input_dir,"kegg_compounds.csv"))  
metaCycDict =  
getMetaCycCompound(os.path.join(input_dir,"metacyc_compounds.csv"))  
ecGenesDict =  
getOptGenes(os.path.join(input_dir,"2012_optGenes_ecoHost_bothDB.txt"))
```

```
print "Still writing all results in output folder (html-file) !!!!!"
```

```
writeResultFiles(maxIteration,bkmReactDict,keggDict,metaCycDict,ecGenesDict  
)
```

```
print "Finished all nonnative metabolites and Heterologous reactions in  
output folder !!!!!!!"
```

```
#### To create index.html file that summarized all connectable nonnative  
metabolites #####
```

```
output_dir = os.path.join(cur_dir,"output")  
writeIndexFile(output_dir,keggDict,metaCycDict)
```

```
print "#####==End=#####"
```

A.1.2 “heterologousReactionDetection.py”

```
#!/usr/bin/env python
# Author: Sunisa Chatsurachai
# Purpose: For finding all additional reactions (Heterologous pathway) for
# each non-native metabolite
# Created: 11/26/2010

import sys,pickle,re,os,csv,glob
from candidateReaction import *
from createHeterologousPathwayFile import writeAllHeterologousPathwayToFile

#####
class Record:
    """Blank class"""
    pass

#####

class Reaction():
    """
    #-----
    def __init__(self,reaction_file):
        """
        reaction_file = "finalBKM_ir.csv"
        EX:
        R00004 2.3.1.48B5407080acetyl-CoA + c-Myc <=> CoA + acetylated c-
Myc
        1 C00024 + 1 c-Myc <=> 1 C00010 + 1 acetylated c-Myc  Brenda_r
        """
        self.reaction_file = open(reaction_file)
        self.data = []
        self.process()
    def getReaction(self):
        """
        get each reaction from file
        """
        return self.data

    def checkGlycanReaction(self,list_equation_item,default = "No glycan"):
        for item in list_equation_item:
            if item.startswith("G"):
                print "found Glycan"
        ##
```

```
        default = None
        break
    return default

def checkMetabolite(self,item):
    """
    Check if it is metabolite or not
    if not return None
    """
    if item in ["<=>", "<-->", "<=", "<--", "=", "-->", "+"]:
        return None

    elif re.findall(r'[a-z]|[A-Z]',item) == []:## found only numeric
        return None
    elif item != "":
        return None

    else:
        return item

def getMetabolitesFromEquation(self,equation):
    """
    input equation
    return [leftmets,rightmets]
    """
    result = []
    pattern = re.search(r'(\<=\>)|(\<=\>)|(\<\-\-\>)|(\<\-\>)\>',equation)

    if pattern is not None:
        #print pattern.group()
        data = equation.split(pattern.group())
        leftMet = data[0]
        rightMet = data[1]
        #print leftMet,rightMet

        result.append(leftMet)
        result.append(rightMet)

    else:
        print "not found direction"
        return result

def getLeftMetabolite(self,leftMets):
```

```

result = []
metItem = leftMets.lstrip().rstrip().split(" + ")

for m in metItem:
    m_item = m.split()

    if self.checkMetabolite(m_item[0]) is None and
re.findall(r'^[1-9]$\^[1-9][0-9]$\^[1-9][0-9][0-9]$',m_item[0]) != []:
        #print m_item[0], "##", " ".join(m_item[1:])
        stoi = m_item[0]
        met = " ".join(m_item[1:])
    else:
        if re.findall(r'^n$|^m$|^(n\+[1-9]\)|^(n\+m\)|^(m\+[1-
9]\)|$)',m_item[0]) != []:
            #print m_item[0], "##", " ".join(m_item[1:])
            stoi = '1'
            met = " ".join(m_item[1:])
        elif re.findall(r'^[2-9]n$|^[2-9]m$|^[1-9][0-9]n$|^[1-9][0-
9]m$',m_item[0]) != []:
            stoiData = m_item[0]
            if len(stoiData) >= 2:
                stoi = stoiData[:-1]
                met = " ".join(m_item[1:])
            else:
                stoi = '1'
                met = " ".join(m_item[:])
        #print stoi,met
        result.append(met)
    return result

def getRighthMetabolite(self,rightMets):
    result = []
    metItem = rightMets.lstrip().rstrip().split(" + ")

    for m in metItem:
        m_item = m.split()
        if self.checkMetabolite(m_item[0]) is None and
re.findall(r'^[1-9]$\^[1-9][0-9]$\^[1-9][0-9][0-9]$',m_item[0]) != []:
            #print m_item[0], "##", " ".join(m_item[1:])
            stoi = m_item[0]
            met = " ".join(m_item[1:])
        else:

```

```

            if re.findall(r'^n$|^m$|^(n\+[1-9]\)|^(n\+m\)|^(m\+[1-
9]\)|$)',m_item[0]) != []:
                #print m_item[0], "##", " ".join(m_item[1:])
                stoi = '1'
                met = " ".join(m_item[1:])
            elif re.findall(r'^[2-9]n$|^[2-9]m$|^[1-9][0-9]n$|^[1-9][0-
9]m$',m_item[0]) != []:
                stoiData = m_item[0]
                if len(stoiData) >= 2:
                    stoi = stoiData[:-1]
                    met = " ".join(m_item[1:])
                else:
                    stoi = '1'
                    met = " ".join(m_item[:])
            result.append(met)
        return result

    def process(self):
        for rx_id,ecnumbers,bkm_id,org_equation,equation_used,revDatabase
in csv.reader(self.reaction_file,delimiter="\t"):
            if self.checkGlycanReaction(equation_used.split()) == None:##
No process if found glycan metabolite
                continue
            else:

                rRecord = Record()
                rRecord.rx_id = rx_id
                rRecord.enzyme = ecnumbers
                rRecord.equation = equation_used
                rRecord.orgEquation = org_equation
                preLeft,preRight =
self.getMetabolitesFromEquation(equation_used)
                rRecord.leftMet = self.getLeftMetabolite(preLeft)
                rRecord.rightMet = self.getRighthMetabolite(preRight)
                preDirection = revDatabase.split("_")[-1]
                if preDirection == "ir":
                    rRecord.direction = "=>"
                else:
                    rRecord.direction = "<=>"

                self.data.append(rRecord)

```

```

def writePreResult(maxIt, nonNativeMetList, kegg_rns, gemMetList):
    cur_dir = os.getcwd()
    for it in range(1, int(maxIt)+1):

        if it == 1: ## 1st searching
            result = getSingleAddedRn(nonNativeMetList, kegg_rns, gemMetList)
            set_r1 = [] ## keep reaction used to connect to host GEM
            set_c1 = [] ## keep non native compound found connecting
            f =
            open(os.path.join(cur_dir, "preResult_iteration_"+str(it)+'.csv'), 'w')
            for cpd, list_rn, min_rn in result:
                set_r1.extend(list_rn)
                set_c1.append(cpd)
                write_result = createListOfAddedRns(cpd, list_rn, it)
                for rec in write_result:
                    f.write("\t".join(map(str, rec)))
                    f.write("\n")
            f.close()

            pre_Rn = set_r1
            pre_Cpd = set(set_c1) ## remove compound duplicate (for next
iteration)

            nextNonNativeMetList =
            list(set(nonNativeMetList).difference(pre_Cpd))
            print "Iteration = ", it, "\t", "No. of connected metabolites =
            ", len(pre_Cpd), "\t", "No. of nonnative metabolites for next step =
            ", "\t", len(nextNonNativeMetList)

            elif it == 2:
                second_result =
                getAddedRns(pre_Rn, list(pre_Cpd), gemMetList, nextNonNativeMetList, kegg_rns, i
t)

                set_r2 = []
                set_c2 = []
                f =
                open(os.path.join(cur_dir, "preResult_iteration_"+str(it)+'.csv'), 'w')
                for cpd, list_rn, min_rn in second_result:
                    set_r2.extend(list_rn)
                    set_c2.append(cpd)
                    write_result = createListOfAddedRns(cpd, list_rn, it)
                    for rec in write_result:
                        f.write("\t".join(map(str, rec)))
                        f.write("\n")
                f.close()

                pre_Cpd = set(pre_Cpd)
                nextNonNativeMetList =
                list(set(nextNonNativeMetList).difference(pre_Cpd))
                print "Iteration = ", it, "\t", "No. of connected metabolites =
                ", len(pre_Cpd), "\t", "No. of nonnative metabolites for next step =
                ", "\t", len(nextNonNativeMetList)
                if len(pre_Cpd) == 0:
                    break
                return it-1 ## return maximum iteration that found connecting nonnative
metabolite

            write_result = createListOfAddedRns(cpd, list_rn, it)
            for rec in write_result:
                f.write("\t".join(map(str, rec)))
                f.write("\n")
            f.close()
            set_c2 = set(set_c2)

            pre_Cpd = set(set_c2)
            pre_Rn = set_r2
            nextNonNativeMetList =
            list(set(nextNonNativeMetList).difference(pre_Cpd))

            print "Iteration = ", it, "\t", "No. of connected metabolites =
            ", len(pre_Cpd), "\t", "No. of nonnative metabolites for next step =
            ", "\t", len(nextNonNativeMetList)

            else:
                next_result =
                getAddedRns(pre_Rn, list(pre_Cpd), gemMetList, nextNonNativeMetList, kegg_rns, i
t)

                pre_Rn = []
                pre_Cpd = []
                f =
                open(os.path.join(cur_dir, "preResult_iteration_"+str(it)+'.csv'), 'w')
                for cpd, list_rn, min_rn in next_result:
                    pre_Rn.extend(list_rn)
                    pre_Cpd.append(cpd)
                    write_result = createListOfAddedRns(cpd, list_rn, it)
                    for rec in write_result:
                        f.write("\t".join(map(str, rec)))
                        f.write("\n")
                f.close()

                pre_Cpd = set(pre_Cpd)
                nextNonNativeMetList =
                list(set(nextNonNativeMetList).difference(pre_Cpd))
                print "Iteration = ", it, "\t", "No. of connected metabolites =
                ", len(pre_Cpd), "\t", "No. of nonnative metabolites for next step =
                ", "\t", len(nextNonNativeMetList)
                if len(pre_Cpd) == 0:
                    break
                return it-1 ## return maximum iteration that found connecting nonnative
metabolite

```

```
def removePreResultFile():
    import glob
    cur_dir = os.getcwd()
    preResultFiles = glob.glob(os.path.join(cur_dir, "*.csv"))
    for f in preResultFiles:
        os.remove(f)

def main():
    pass

if __name__ == '__main__': main()
```

A.1.3 “candidateReaction.py”

```
#!/usr/bin/env python
# Author: Sunisa Chatsurachai
# Purpose: functions for finding candidate reaction(s)
# Created: 11/26/2010

import os, re, sys
from glob import glob

def allPrestepsMapping(preStepPath, can_rxn_metList):
    ## preStepPath =
    "D:\\SUNISA_data\\2012_additional_reaction\\2012JUL_bkmReact\\2012JUL_addRe
action\\assume_r\\eco\\"
    metNotFound = []
    connectMets = []
    metFound = []
    for met in can_rxn_metList:
        list_csv = glob("*.csv") ## list of preResult File
        for f in list_csv:
            for line in open(f, 'r'):
                if line.startswith(met) == True:
                    metFound.append(met)
                    break
    metNotFound = set(can_rxn_metList).difference(set(metFound))
    return [list(metNotFound), metFound]

def hostMetaboliteMapping(gemMetList, can_rn_metList):
    """
    Comparing metabolite from candidate reaction with genome-scale
    model(GEM) metabolite list
    and return list of metabolite does not found in GEM list
    """
    met_not_found = []
    for item in can_rn_metList:
        if item not in gemMetList:
            met_not_found.append(item)
    return list(set(met_not_found))

def findAllPossibleCandidateRn(kegg_rns, nonNativeMet, gemMetList):
    """
```



```

kegg_rns (all metabolic reaction from KEGG (no glycan reaction here)
Type is Dictionary
each record as
key -> item.rx_id
value list->
[item.rx_id,item.leftMet,item.rightMet,item.equation,item.equation_name,item
m.direction]
"""
pre_candidate_rn = []
for key,value in kegg_rns.iteritems():
    rx_id = value[0]
    leftMet = value[1]
    rightMet = value[2]
    direction = value[-1]

    if direction == "<=>": ## found reversible reaction
        if nonNativeMet in leftMet:
            rightMet = hostMetaboliteMapping(gemMetList,rightMet)##
compare and return met not found in GEM
            pre_candidate_rn.append([rx_id,rightMet])
        elif nonNativeMet in rightMet:
            leftMet = hostMetaboliteMapping(gemMetList,leftMet)
            pre_candidate_rn.append([rx_id,leftMet])
        else:## found irreversible reaction
            if nonNativeMet in rightMet:
                leftMet = hostMetaboliteMapping(gemMetList,leftMet)
                pre_candidate_rn.append([rx_id,leftMet])

result = {}
for k,m in pre_candidate_rn:
    result[k] = m
return result

def getSingleAddedRn(nonNativeMetList,kegg_rns,gemMetList):
    """
    At 1st iteration -> finding non native metabolite which can connect to
host by adding single reaction
    """
    result ={}
    final_result = []
    for nonNativeMet in nonNativeMetList:

```

```

        eachNonNativeMet =
findAllPossibleCandidateRn(kegg_rns,nonNativeMet,gemMetList)
        if [] in eachNonNativeMet.itervalues():## [] found in possible
pathway of connected metabolite
            candidate_rn = []
            for key,val in eachNonNativeMet.iteritems():
                if val == []: ##all substrate found in GEM list
                    candidate_rn.append(key) ## append reaction id
            result[nonNativeMet] = candidate_rn
        for k,v in result.iteritems():
            final_result.append([k,v,len(v)]) ## create list of all possible
reaction
    return final_result

def
getAddedRns(preStepRn,preStepMet,gemMetList,nonNativeMetList,kegg_rns,itera
tion):
    """
    From 2nd iteration -> finding at least two reactions to be added to
host for connecting non-native metabolite
    3rd iteration -> finding at least three reactions to be added to
host for connecting non-native metabolite
    ....
    """
    next_step_result = []

    for nonNativeMet in nonNativeMetList:
        pre_rn =
findAllPossibleCandidateRn(kegg_rns,nonNativeMet,gemMetList)
        pre_result = {}
        possible_rns = []

        for rx_id,met_not_found in pre_rn.iteritems():
            min_rn_found = 0+iteration ## at least start from iteration
number ( to connect to host metabolite)

            if len(met_not_found) == 1:## only on metabolite does not found
in GEM list
                if met_not_found[0] in preStepMet:### found in previous
step added reaction (iteration -1)
                    ## found connecting by adding one more reaction (from
previous step)
                    min_rn_found = min_rn_found

```

```

possible_rns.append([rx_id,met_not_found[0],min_rn_found])
    else:

metNotFoundInAllPresteps,connectedMets=allPrestepsMapping(os.getcwd(),met_not_found)

    if metNotFoundInAllPresteps == []:## all metabolite
found in presteps
        min_rn_found = min_rn_found+len(met_not_found)
        preMets = []
        for cMet in connectedMets:
            cur_cMet = cMet.split("_")[0]
            listReaction = re.findall("R[0-9][0-9][0-9][0-9][0-9]",cMet)

            preMets.append(cur_cMet)
            preMets = list(set(preMets))

possible_rns.append([rx_id,preMets[0],min_rn_found])
    else:## more than one metabolite does not found in GEM list
        met_found_in_preStep = []
        for i,m in enumerate(met_not_found):
            if m in preStepMet:
                met_found_in_preStep.append(m)

        cur_metNotFound =
set(met_not_found).difference(set(met_found_in_preStep))
        cur_metNotFound = list(cur_metNotFound)

        if cur_metNotFound == []:## all metabolites already found
in prestep metabolite
            min_rn_found = min_rn_found +len(met_not_found)## at
least reaction should be added

possible_rns.append([rx_id,met_found_in_preStep,min_rn_found])

    else:

metNotFoundInAllPresteps,connectedMets=allPrestepsMapping(os.getcwd(),cur_metNotFound)

    if metNotFoundInAllPresteps == []:## all metabolite
found in presteps
        min_rn_found = min_rn_found+len(met_not_found)

```

```

preMets = []
for cMet in connectedMets:
    cur_cMet = cMet.split("_")[0]
    listReaction = re.findall("R[0-9][0-9][0-9][0-9][0-9]",cMet)

    preMets.append(cur_cMet)

for m in met_found_in_preStep:
    preMets.append(m)

preMets = list(set(preMets))
possible_rns.append([rx_id,preMets,min_rn_found])

pre_result[nonNativeMet] = possible_rns
if pre_result[nonNativeMet] != []:

next_step_result.append([nonNativeMet,possible_rns,len(possible_rns)])

return next_step_result

def createListOfAddedRns(cpd_id,ListRns,iteration):
    rec = []
    if iteration ==1: ## 1st iteration result
        if len(ListRns) == 1:
            rn = ListRns[0]
            rec.append([cpd_id,rn,iteration])
        else:
            for rn in ListRns:
                rec.append([cpd_id,rn,iteration])
    return rec
else:
    ## ListRns=[['R01424', 'C00180']]
    if len(ListRns) == 1:
        rn = ListRns[0][0] ## rx_id
        if rn.find("##") != -1:
            rn = rns.split('##')
        else:
            rn
            cpd_pre = ListRns[0][1] ## compound id
            min_rn_found = ListRns[0][2]

            if type(cpd_pre) != list:## only one previous step metabolite
(in iteration -1)

```

```

        rec.append([cpd_id,rn,iteration,cpd_pre,min_rn_found])
    else:
        rec.append([cpd_id,rn,iteration,cpd_pre,min_rn_found])

else:
    ## [['R07922', 'C16353'], ['R07921', 'C07481'], ['R07954',
    'C07481'], ['R07939', 'C07481'], ['R07943', 'C16358']]
    ## [['R03186', ['C00072', 'C05422']], ['R00095', 'C00072']]
    for r in ListRns:
        rn = r[0]
        if rn.find('##') != -1:
            rn = rn.split('##')
        else:
            rn

        cpd_pre = r[1]
        min_rn_found = r[-1]

        if type(cpd_pre) != list:
            rec.append([cpd_id,rn,iteration,cpd_pre,min_rn_found])
        else:
            rec.append([cpd_id,rn,iteration,cpd_pre,min_rn_found])
    return rec

```

A.1.4 “createHeterologousPathwayFile.py”

```

#!/usr/bin/env python
# Author: Sunisa Chatsurachai
# Purpose: Create additional pathway detection for simulation (txt-file)
# Created: 11/29/2010

```

```

import sys,os,csv,re
from glob import glob

```

```

def addCompartmentToMet(reaction,compartment="[c]"):
    result = []
    reaction = reaction.upper().split()
    for item in reaction:
        if item not in ["<=", "<=", ">=", ">=", "+"] and re.findall(r'[0-9]',item[0]) == []: ## only metabolite
            item = item+compartment
            result.append(item)
        else:
            result.append(item)

    return result

```

```

def addTargetMetOutputRn(nonNativeMet):
    met_ext = nonNativeMet.upper()+"[e]"
    sink_met_rn = " ".join([met_ext,'=>'])
    return sink_met_rn

```

```

def createTransportRn(nonNativeMet):
    met_cytosol = nonNativeMet.upper()+"[c]"
    met_extracellular = nonNativeMet.upper()+"[e]"
    transport_rn = " ".join([met_cytosol,"=>",met_extracellular])
    return transport_rn

```

```

def getPreviousStepRn(preMet,pre_it):
    """
    get all reactions of previous metabolite
    """
    cur_dir = os.getcwd()
    for iteration in range(pre_it):

```

```

targetDir = os.path.join(cur_dir,"iteration_"+str(iteration+1))

fileList = glob(targetDir+"/*.txt")
results = []
for fName in fileList:
    file_name = fName.split("\\")[-1]
    met_name = file_name.split("_")[0]
    if met_name == preMet:
        rns = [line.rstrip() for line in open(fName)]
        previous_rns_name = rns[0]
        list_rns = rns[1:-2]
        results.append([previous_rns_name,list_rns])
    if results != []:
        break
return results

def getAllPossiblePreviousStepRn(preMetList,cur_it):
    """
    get all reaction of all previous metabolites(as list)
    """
    preMetList_rns = {}
    for preMet in preMetList:
        rns = []
        pre_rn_set = getPreviousStepRn(preMet,cur_it)
        if pre_rn_set != []:
            if len(pre_rn_set) == 1:## one possible set of reaction
                pre_cid_rid = pre_rn_set[0][0]
                pre_list_rn = pre_rn_set[0][1][:]
                rns.append([pre_cid_rid,pre_list_rn])
            else:
                for pre_cid_rid, pre_list_rn in pre_rn_set:
                    rns.append([pre_cid_rid,pre_list_rn])
        preMetList_rns[preMet] =rns
    return preMetList_rns

def changeFormatPreviousListMet(preMet):
    ## nonNativeMet addedRn iteartion previousMets minimum_rn
    ##C16688 R03921 2 ['C02336', 'C00668'] 4##
    preMetList = ""
    for char in preMet:
        if char not in ["[","]","'",","," "]:
            preMetList =preMetList +char
    preMetList = sorted(preMetList.split(","))

```

```

return preMetList

def writeResultToTextFile(targetDirectory,iteration,List_result):
    ## List_result =
    [cur_rn_name,pre_rn_name,cur_rn,pre_rn,cur_tran_rn,sink_met_rn,path_no]

    if iteration > 2:

        f =
        open(targetDirectory+str(iteration)+"/"+List_result[0]+"_"+str(List_result[
-1]))+".txt", "w")
        f.write(List_result[0]+"_"+List_result[1])
        f.write("\n")
        if len(List_result[3]) == 1: # only one heterologous pathway
            f.write(List_result[3][0])
            f.write("\n")
        else:
            rns = List_result[3]
            for r in rns:
                f.write(r)
                f.write("\n")
            f.write("\n".join([List_result[2],List_result[4],List_result[5]]))
            f.write("\n")

    else:
        f =
        open(targetDirectory+str(iteration)+"/"+List_result[0]+"_"+List_result[1]+
.txt", "w")
        f.write(List_result[0]+"_"+List_result[1])
        f.write("\n")
        if len(List_result[3]) == 1: # only one heterologous pathway
            f.write(List_result[3][0])
            f.write("\n")
        else:
            rns = List_result[3]
            for r in rns:
                f.write(r)
                f.write("\n")
            f.write("\n".join([List_result[2],List_result[4],List_result[5]]))
            f.write("\n")

def writeAllHeterologousPathwayToFile(maxIt,kegg_rns):

```

```

cur_dir = os.getcwd()
for iteration in range(1,maxIt+2):

    if not
os.path.exists(os.path.join(cur_dir,"iteration_"+str(iteration))):
    os.mkdir(os.path.join(cur_dir,"iteration_"+str(iteration)))

    if iteration ==1:
        f =
open(os.path.join(cur_dir,"preResult_iteration_"+str(iteration)+".csv"),"r"
)

        for nonNativeMet,r_id,it in csv.reader(f,delimiter= "\t"):
            if kegg_rns.has_key(r_id):
                eq_with_compartment = "
".join(addCompartmentToMet(kegg_rns[r_id][3],[c]))
                add_rn = [nonNativeMet,r_id,eq_with_compartment]
                try:
                    f1 =
open(cur_dir+"/iteration_1/"+nonNativeMet+"_"+r_id+".txt",'w')
                    fName = nonNativeMet+"_"+r_id

f1.write("\n".join([fName,eq_with_compartment,createTransportRn(nonNativeMe
t),addTargetMetOutputRn(nonNativeMet)]))
                    f1.write("\n")
                    f1.close()

                except IOError:
                    f1 =
open(cur_dir+"/iteration_1/"+nonNativeMet+"_"+r_id+".txt",'w')
                    fName = nonNativeMet+"_"+r_id

f1.write("\n".join([fName,eq_with_compartment,createTransportRn(nonNativeMe
t),addTargetMetOutputRn(nonNativeMet)]))
                    f1.write("\n")
                    f1.close()

            if iteration > 1 and iteration <= maxIt+2:

                f =
open(os.path.join(cur_dir,"preResult_iteration_"+str(iteration)+".csv"),"r"
)

```

```

        for nonNativeMet,r_id,it,preMet,min_rn in
csv.reader(f,delimiter="\t"):
            path_no = 0
            if int(min_rn) == iteration:## means -> connecting by one
to one metabolite since other found in GEM List
                cur_rn_with_com = "
".join(addCompartmentToMet(kegg_rns[r_id][3],[c]))
                cur_rn_name = nonNativeMet+"_"+r_id
                cur_trans_rn = createTransportRn(nonNativeMet)
                sink_met_rn = addTargetMetOutputRn(nonNativeMet)
                #pre_rn_set =
getPreviousStepRn(preMet,cur_dir+"/iteration_"+str(iteration-1))
                pre_rn_set = getPreviousStepRn(preMet,iteration-1)

                if len(pre_rn_set)==1: ## only one possible added

pathway

                    path_no = path_no+1
                    pre_cid_rid = pre_rn_set[0][0]
                    pre_list_rn = pre_rn_set[0][1][:]

                    List_result
=[cur_rn_name,pre_cid_rid,cur_rn_with_com,pre_list_rn,cur_trans_rn,sink_met
_rn,path_no]

writeResultToTextFile(cur_dir+"/iteration_",iteration,List_result)

                else:
                    for pre_cid_rid,pre_list_rn in pre_rn_set:
                        path_no = path_no+1
                        List_result
=[cur_rn_name,pre_cid_rid,cur_rn_with_com,pre_list_rn,cur_trans_rn,sink_met
_rn,path_no]

writeResultToTextFile(cur_dir+"/iteration_",iteration,List_result)

            else:
                cur_rn_with_com = "
".join(addCompartmentToMet(kegg_rns[r_id][3],[c]))
                cur_rn_name = nonNativeMet+"_"+r_id
                cur_trans_rn = createTransportRn(nonNativeMet)
                sink_met_rn = addTargetMetOutputRn(nonNativeMet)

```

```

        pre_metList = changeFormatPreviousListMet(preMet)
        pre_metListRn =
getAllPossiblePreviousStepRn(pre_metList,iteration-1)

        first_substrate_rn = pre_metListRn[pre_metList[0]]
        del pre_metListRn[pre_metList[0]] ## update list of
previous step metabolite

        for p_rn in first_substrate_rn:
            pre_cid_rid = p_rn[0]
            pre_list_rn = p_rn[1][:]

            for key,value in pre_metListRn.iteritems():
                if len(value)== 1: ##only one possible pathway
                    ##key = C02557
                    ##value = [['C02557_R05373', ['C02557[c]
<=> C00100[c] + C00011[c]']]
                    path_no = path_no+1
                    new_pre_list_rn = []
                    v_cid_rid = value[0][0]
                    v_list_rn = value[0][1]
                    new_pre_list_rn.extend(pre_list_rn)
                    new_pre_list_rn.extend(v_list_rn)

                    List_result
=[cur_rn_name,pre_cid_rid+"_"+v_cid_rid,cur_rn_with_com,new_pre_list_rn,cur
_trans_rn,sink_met_rn,path_no]

writeResultToTextFile(cur_dir+"/iteration_",iteration,List_result)
                else:
                    for v_cid_rid,v_list_rn in value:
                        new_pre_list_rn = []
                        path_no = path_no+1
                        new_pre_list_rn.extend(pre_list_rn)
                        new_pre_list_rn.extend(v_list_rn)
                        List_result
=[cur_rn_name,pre_cid_rid+"_"+v_cid_rid,cur_rn_with_com,new_pre_list_rn,cur
_trans_rn,sink_met_rn,path_no]

writeResultToTextFile(cur_dir+"/iteration_",iteration,List_result)

def main():
    pass

```

```

if __name__=='__main__':main()

```

A.1.5 “addInfoOutput.py”

```
#!/usr/bin/env python
# Author: Sunisa Chatsurachai --<>
# Purpose: To add more information of the output files
# Created: 27-Feb-13
```

```
import sys,os,re
import glob
from heterologousReactionDetection import *
```

```
def getKEGGcompound(fName):
    result = {}
    for line in open(fName,'r'):
        line = line.lstrip().rstrip().split('\t')
        cpd_id = line[0]
        cpd_syn = line[1]
        cpd_common = line[2]
        cpd_charge = line[3]
        cpd_formula = line[4]
        result[cpd_id] = [cpd_syn,cpd_formula]
    return result
```

```
def getMetaCycCompound(fName):
    result = {}
    for line in open(fName,'r'):
        line = line.lstrip().rstrip().split('\t')
        m_id = line[0]
        k_id = line[1]
        cpd_syn = "###".join([line[2],line[3]])
        cpd_charge = line[5]
        cpd_formula = line[6]
        result[m_id] = [cpd_syn,cpd_formula,k_id]
    return result
```

```
def getBKMrreaction(fName):
    result = {}
    for line in open(fName,'r'):
        line = line.lstrip().rstrip().split('\t')
        r_id = line[0]
        ec_num = line[1]
```

```
        refID = line[2]
        eq_name = line[3]
        eq = line[4]
        type_rxn = line[5]
        result[r_id] = [eq_name,eq,ec_num,refID,type_rxn]
    return result
```

```
def getReactions(listReactions,bkmDict):
    rxns = listReactions.split('##')
    result = {}
    for r in rxns:
        eqName,eq,ec_id,refID,revData = bkmDict[r]
        result[r] = eqName
    return result
```

```
def getOptGenes(fName):
    result = {}
    for line in open(fName,'r'):
        line = line.lstrip().rstrip().split('\t')
        ec = line[0]
        #db = line[1]
        #genes = line[2]
        #orfList = line[3]
        #org_id = line[4]
        #org_name = line[5]
        result[ec] = line[1:6]
    return result
```

```
def getCompoundName(target,keggDict,MetacycDict):
    if keggDict.has_key(target):
        cpd_syn,cpd_formula = keggDict[target]
    elif MetacycDict.has_key(target):
        cpd_syn,cpd_formula,k_id = MetacycDict[target]
    else:
        cpd_syn = "None"
        cpd_formula = "None"
    return [prepareCompoundName(cpd_syn),cpd_formula]
```

```
def prepareCompoundName(cpdName):
    cpdName = cpdName.split("###")
    cpdName = set(cpdName)
    cpdName = list(cpdName)
```

```

if len(cpdName)==1:
    met = cpdName[0]
    if met.upper() == "NONE" or met.upper() == "NULL":
        return "No information"
    else:
        return met
else:
    data = []
    for m in cpdName:
        if m.upper() != "NONE" and m.upper()!="NULL":
            data.append(m)
    return "||".join(data)

def
writeResultFiles(maxIteration,bkmReactDict,keggDict,metaCycDict,ecGenesDict
):
    cur_dir = os.getcwd()

    if os.path.exists(os.path.join(cur_dir,"output")) == False:
        os.mkdir(os.path.join(cur_dir,"output"))

    for it in range(1,maxIteration+1+1):
        curRentPath = os.path.join(os.getcwd(),"iteration_"+str(it))
        listTextfiles = glob(curRentPath+"/*.txt")

        outputDir = os.path.join(cur_dir,"output")

        if os.path.exists(outputDir)== False:
            os.mkdir(outputDir)

        for f in listTextfiles:
            f_op = open(f,'r')
            line = f_op.readline()
            f_op.close()

            foutName = f.split("\\")[-1][:-4]

            connMet= line.lstrip().rstrip().split("_")
            targetName,targetFormula =
getCompoundName(connMet[0],keggDict,metaCycDict)

            outputFile = os.path.join(outputDir,foutName+".html")

```

```

f_out = open(outputFile,'w')
f_out.write('<html>\n')
f_out.write('<body>\n')
f_out.write("&table border = '2'>\n")
f_out.write('<h1> Result </h1>\n')
f_out.write('<tr>\n')
f_out.write('<td> Target compound ID </td>\n')
f_out.write('<td>'+connMet[0]+'</td>\n')
f_out.write('</tr>\n')

f_out.write('<tr>\n')
f_out.write('<td> Target compound Name </td>\n')
f_out.write('<td>'+targetName+'</td>\n')
f_out.write('</tr>\n')

listRxns = re.findall(r'R[0-9][0-9][0-9][0-9][0-9]',line)

f_out.write('<tr>\n')
f_out.write('<td> List of Heterologous reaction(s) </td>\n')
f_out.write('<td>'+"".join(listRxns)+'</td>\n')
f_out.write('</tr>\n')

for r_id in listRxns:
    f_out.write('<tr>\n')
    f_out.write('<td> Detail of heterologous reaction</td>\n')

    r_data = bkmReactDict[r_id]
    eq_name = r_data[0]
    eq_id = r_data[1]
    ec_id = r_data[2]

    f_out.write('<td>'+r_id+'<br>'+eq_name+'<br>'+eq_id+'<br>')

    listEC = ec_id.split("###")
    if len(listEC) ==1: ## found only one EC number

        try:
            e = listEC[0]
            data_hetGenes = ecGenesDict[e]
            db = data_hetGenes[0]
            geneSym = data_hetGenes[1]
            orfList = data_hetGenes[2]
            org_id = data_hetGenes[3]

```



```

        org_name = data_hetGenes[4]
        f_out.write( "-----"
        -----"+<br>'+Enzyme information:')
        f_out.write('<br>'+e+'<br>'+Gene symbol:
'+geneSym.replace("##",";")+<br>'+ORF:  "+orfList.replace("##","|")+
from database  '+db+'<br>'+Organism:  '+org_name+'<br>')
        f_out.write('</td>\n')

    except KeyError,err:
        f_out.write( "-----"
        -----"+<br>'+Enzyme information:')
        f_out.write('<br>'+listEC[0]+'<br>'+No information
about genes')
        f_out.write('</td>\n')

    else:
        f_out.write( "-----"
        -----"+<br>'+Enzyme information:')
        for e in listEC:
            try:
                data_hetGenes = ecGenesDict[e]
                db = data_hetGenes[0]
                geneSym = data_hetGenes[1]
                orfList = data_hetGenes[2]
                org_id = data_hetGenes[3]
                org_name = data_hetGenes[4]
                f_out.write('<br>'+e+'<br>'+Gene symbol:
'+geneSym.replace("##",";")+<br>'+ORF:  "+orfList.replace("##","|")+
from database  '+db+'<br>'+Organism:  '+org_name+'<br>')

            except KeyError,err:
                f_out.write('<br>'+e+'<br>'+No information
about genes'+<br>')

        f_out.write('_____')

        f_out.write('</td>\n')

    f_out.write('</tr>\n')

```

```

f_out.write('<footer>')
f_out.write('<p>Created by: Sunisa Chatsurachai</p>')
f_out.write('<p>Date: 2012-02-28</p>')
f_out.write('</footer>\n')
f_out.write('</table>\n')
f_out.write('</body>\n')
f_out.write('</html>\n')
f_out.close()

```

```
def main():
```

```

    cur_dir = os.getcwd()
    input_dir = os.path.join(cur_dir,"input")

```

```

    bkmReactDict =
getBKMreaction(os.path.join(input_dir,"bkm_metabolic_reactions_2012.csv"))
    keggDict =
getKEGGcompound(os.path.join(input_dir,"kegg_compounds.csv"))
    metaCycDict =
getMetaCycCompound(os.path.join(input_dir,"metacyc_compounds.csv"))
    ecGenesDict =
getOptGenes(os.path.join(input_dir,"2012_optGenes_ecoHost_bothDB.txt"))

```

```
    maxIteration = 3
```

```

writeResultFiles(maxIteration,bkmReactDict,keggDict,metaCycDict,ecGenesDict
)

```

```
if __name__=='__main__':main()
```

A.1.6 “createIndexFile.py”

```
#!/usr/bin/env python
# Author: Sunisa Chatsurahcai  --<>
# Purpose: To create index.html for link result of nonnative metabolites
# Created: 28-Feb-13
```

```
import sys,os,re
```

```
def getKEGGcompound(fName):
    result = {}
    for line in open(fName,'r'):
        line = line.lstrip().rstrip().split('\t')
        cpd_id = line[0]
        cpd_syn = line[1]
        cpd_common = line[2]
        cpd_charge = line[3]
        cpd_formula = line[4]
        result[cpd_id] = [cpd_syn,cpd_formula]
    return result

def getMetaCycCompound(fName):
    result = {}
    for line in open(fName,'r'):
        line = line.lstrip().rstrip().split('\t')
        m_id = line[0]
        k_id = line[1]
        cpd_syn = "###".join([line[2],line[3]])
        cpd_charge = line[5]
        cpd_formula = line[6]
        result[m_id] = [cpd_syn,cpd_formula,k_id]
    return result

def getCompoundName(target,keggDict,MetacycDict):
    if keggDict.has_key(target):
        cpd_syn,cpd_formula = keggDict[target]
    elif MetacycDict.has_key(target):
        cpd_syn,cpd_formula,k_id = MetacycDict[target]
    else:
        cpd_syn = "None"
        cpd_formula = "None"
    return [prepareCompoundName(cpd_syn),cpd_formula]
```

```
def prepareCompoundName(cpdName):
    cpdName = cpdName.split("###")
    cpdName = set(cpdName)
    cpdName = list(cpdName)
    if len(cpdName)==1:
        met = cpdName[0]
        if met.upper() == "NONE" or met.upper() == "NULL":
            return "No information"
        else:
            return met
    else:
        data = []
        for m in cpdName:
            if m.upper() != "NONE" and m.upper()!="NULL":
                data.append(m)
        return "||".join(data)

def getAllConnectedMetaboites(targetDir):
    import glob
    allHtmls = glob.glob(os.path.join(targetDir+"/*.html"))
    allTargets = []
    for f in allHtmls:
        target = f.split("\\")[1]
        target = target.split("_")[0]
        allTargets.append(target)
    return list(set(sorted(allTargets)))

def getHtmlLinks(cpdId,targetDir):### limited to 5 possible way only ###
    import glob,random
    htmls = glob.glob(os.path.join(targetDir,"%s*.html"%cpdId))

    if len(htmls) < 5:
        targetFiles = htmls[:]
    else:
        targetFiles = []
        targetFiles.append(random.choice(htmls))
        targetFiles.append(random.choice(htmls))
        targetFiles.append(random.choice(htmls))
        targetFiles.append(random.choice(htmls))
        targetFiles.append(random.choice(htmls))
    return targetFiles
```

```

def writeIndexFile(output_dir,keggDict,metaCycDict):
    import os
    cur_dir = os.getcwd()
    allNonMets = getAllConnectedMetaboites(output_dir)
    indexFile = os.path.join(cur_dir,"index.html")
    fopen = open(indexFile,'w')
    fopen.write('<html>\n')
    fopen.write('<h1>'+ 'Index' + '</h1>\n')

    fopen.write('<body>\n')
    fopen.write("&<table border='2'>\n")
    fopen.write('<tr>\n')
    fopen.write('<th>Nonnative metabolite </th>')
    fopen.write('<th>Heterologous pathway(s)</th>')
    fopen.write('</tr>\n')

    for m in allNonMets[:]:

        targetName,targetFormula = getCompoundName(m,keggDict,metaCycDict)
        htmls = getHtmlLinks(m,output_dir)

        fopen.write('<tr>\n')
        fopen.write('<td>'+targetName+" || "+m+'</td>')

        for i,htm in enumerate(htmls):

            targetFile = htm.split("\\")[-1]### name of html file
            fopen.write('<td>')
            fopen.write('<a')
href='+"output\\"'+targetFile+'>'+ "Pathway_"+str(i+1)+'</a>'+ '<br>'
            fopen.write('</td>')
            #<a href="filename.html">text that responds to link</a>
            #fopen.write('<td></td>')
            fopen.write('</tr>\n')

    fopen.write('<footer>')
    fopen.write('<p>Created by: Sunisa Chatsurachai</p>')
    fopen.write('<p>Date generated: 2012-02-28</p>')
    fopen.write('</footer>\n')

```

```

fopen.write("</table>\n")
fopen.write('</body>\n')
fopen.write('</html>\n')
fopen.close()

```

```

def main():

    cur_dir = os.getcwd()
    input_dir = os.path.join(cur_dir,"input")
    keggDict =
getKEGGcompound(os.path.join(input_dir,"kegg_compounds.csv"))
    metaCycDict =
getMetaCycCompound(os.path.join(input_dir,"metacyc_compounds.csv"))

    output_dir = os.path.join(cur_dir,"output")
    writeIndexFile(output_dir,keggDict,metaCycDict)

    if __name__=='__main__':main()

```

A.2 Codon Adaptation Index by Python™ and Biopython

A.2.1 “ecoHost_cai.py”

```
# Author:  SUNISA Chatsurachai --<
# Purpose: Calculate CAI when E. coli as the host
# Created: 18-Nov-11

import sys,os,re

from Bio.Seq import Seq
from Bio.SeqRecord import SeqRecord
from Bio.SeqIO import *
from Bio.SeqUtils.CodonUsage import *
from Bio.SeqUtils.CodonUsageIndices import *
from glob import glob
aminoDict = {'ala':['GCA','GCC','GCG','GCT'],
             'arg':['AGA','AGG','CGA','CGC','CGG','CGT'],
             'asn':['AAC','AAT'],
             'asp':['GAC','GAT'],
             'cys':['TGC','TGT'],
             'gln':['CAA','CAG'],
             'glu':['GAA','GAG'],
             'gly':['GGA','GGC','GGG','GGT'],
             'his':['CAC','CAT'],
             'ile':['ATA','ATC','ATT'],
             'lys':['AAA','AAG'],
             'met':['ATG'],
             'phe':['TTC','TTT'],
             'pro':['CCA','CCC','CCG','CCT'],
             'ser':['AGC','AGT','TCA','TCC','TCG','TCT'],
             'stop':['TAA','TAG','TGA'],
             'thr':['ACA','ACC','ACG','ACT'],
             'trp':['TGG'],
             'tyr':['TAC','TAT'],
             'val':['GTA','GTC','GTG','GTT'],
             'leu':['CTA','CTC','CTG','CTT','TTA','TTG']}

def findKOId(orf_desc):
    orf_desc = orf_desc.split()
```

```
    koID = []
    for item in orf_desc:
        if item.startswith("K") and re.findall("^K[0-9][0-9][0-9][0-9][0-9]$",item) != []:
            koID.append(item)
    if len(koID) >=1 and koID != []:
        return koID
    else:
        return None

def findGenes(orf_desc):
    orf_desc = orf_desc.split(";")[0]
    if orf_desc.find(",") != -1:#found gene information
        print orf_desc

    else:
        pass
def printListRecordsToFile(fName, records, delimiter='\t'):
    f = open(fName,'a')
    f.write(delimiter.join(map(str,records)))
    f.write('\n')
    f.close()

def main():

    cur_dir = os.getcwd()

    heg_ecoli = CodonAdaptationIndex()
    heg_ecoli.generate_index(os.path.join(cur_dir,'heg_ecoli_seq.fasta'))
    ## w value in Sharp's Method
    ##heg_ecoli.print_index()
    #for item,val in heg_ecoli.index.iteritems():
        #print item,val
    #sys.exit()

    kegg_seq_dir =
"D:\\SUNISA_data\\2012_CAI_calculation\\KEGG_cds\\2012JUNE_kegg_cds\\"
    list_seq_file = glob(kegg_seq_dir+"*.csv")

    for fName in list_seq_file:
        org_id = fName.split("\\")[-1].split("_")[0]
```

```

for s in parse(fName, 'fasta'):
    orf_id = s.id[4:]
    orf_desc = s.description
    seq = str(s.seq)
    outputFile =
os.path.join(cur_dir, "output/"+orf_id+"_ecoHost.csv")

    if seq[:3].upper() in ['ATG', 'GTG', 'TTG'] and seq[-3:].upper()
in ['TAA', 'TAG', 'TGA']:
    try:
        s_cai = heg_ecoli.cai_for_gene(seq)
    except TypeError, e:
        print e
        s_cai = "None"

    else:
        print 'NOT CDS', orf_id
        s_cai = "None"

resultGenes = findGenes(orf_desc)
resultKO = findKOId(orf_desc)

if resultKO != None:
    if len(resultKO) == 1:## only one KO id for this orf
        record = [orf_id, orf_desc, len(s.seq), s_cai, org_id]
        ko_id = resultKO[0]
        record.append(ko_id)
        #print record
        printListRecordsToFile(outputFile, record, "\t")
    else:
        for ko_id in resultKO:## one orf has more than one KO
            record = [orf_id, orf_desc, len(s.seq), s_cai, org_id]
            record.append(ko_id)
            #print record
            printListRecordsToFile(outputFile, record, "\t")

    else:
        record = [orf_id, orf_desc, len(s.seq), s_cai, org_id]
        record.append("None")
        printListRecordsToFile(outputFile, record, "\t")

```

```

print "Finished calculation!!!!!!!!!!!!!!!"

if __name__ == '__main__': main()

```

A.2.2 “maxCAIscoreECnumbers_ecoHost.py”

```
# Author: Sunisa Chatsurachai --<>
# Purpose: To collect maximum CAI score for each EC number -> suggest
heterologous gene(s)
# Created: 17-May-12
```

```
import sys,os,re
import glob,time
```

```
def printListToFile(fName,record,delimiter="\t"):
    f = open(fName,'a')
    f.write(delimiter.join(map(str,record)))
    f.write("\n")
    f.close()
```

```
def getECDict(fName):
    result = {}
    for line in open(fName):
        line = line.lstrip().rstrip().split("\t")
        ec_id = line[0]
        ko_ids = line[1].split("##")
        gene_ids = line[2].split("##")
        data = zip(ko_ids,gene_ids)
        data = list(set(data))
        result[ec_id] = data
```

```
    return result
```

```
def checkNoCAI(cai_data):
    if cai_data == "None":
        return None
    else:
        return float(cai_data)
```

```
def getAllGenesFromOrganisms(targetDir,cur_koId):
    caiList = []
    orgList = []
    orfList = []
    for f in glob.glob(targetDir+'/*.csv'):
        for line in open(f):
            line = line.lstrip().rstrip().split("\t")
            cai_data = line[-3]
```

```
            org_id = line[-2]
            ko_id = line[-1]
            orf_id = line[0]
```

```
            if ko_id == cur_koId:
                if checkNoCAI(cai_data) != None:
                    cur_cai = cai_data
                else:
                    cur_cai = 0.00
            caiList.append(cur_cai)
            orgList.append(org_id)
            orfList.append(orf_id)
```

```
    return sorted(set(zip(caiList,orgList,orfList)))
```

```
def getCAIFromAllSubunit(caiDict):
```

```
    allKK = []
    allCai = []
    allGene = []
    for kk,vv in caiDict.iteritems():
        keggID = kk
        caiTuples = vv[0]
        keggGene = vv[1]
        allKK.append(keggID)
        allGene.append(keggGene)
        allCai.extend(caiTuples)
```

```
    orgDict = {}
    for i,v,k in allCai:
        cai = float(i)
        orgID = v
        orfID = k
        if orgDict.has_key(orgID):## already have this organism
            caiData,listOrf = orgDict[orgID]
            caiData.append(cai)
            listOrf.append(orfID)
            orgDict[orgID] = [caiData,listOrf]
```

```
    else:
        orgDict[orgID] = [[cai],[orfID]]
    result = []
    for o, caiOrgs in orgDict.iteritems():
        oId = o
```

```

    caiOrf = "###".join(caiOrgs[1])

    avg_cai = sum(caiOrgs[0],0.0)/len(caiOrgs[0])
    #print oId,avg_cai,caiOrf,"###".join(allKK),"###".join(allGene)
    rec = (avg_cai,oId,caiOrf,"###".join(allKK),"###".join(allGene))
    result.append(rec)
    return list(sorted(set(result)))

def getOrgDict(fName):
    result = {}
    for line in open(fName):
        line = line.lstrip().rstrip().split("\t")
        org_id = line[0]
        org_name = line[1]
        result[org_id] = org_name
    return result

def main():
    start = time.clock()
    cur_dir = os.getcwd()
    targetDir = os.path.join(cur_dir,"output")
    orgNameDict = getOrgDict(os.path.join(cur_dir,"input/orgName.csv"))
    ecDict = getECDict(os.path.join(cur_dir,"input/ecLinkKOIDs.csv"))
    outputFile = os.path.join(cur_dir,"20120CT_maxCAI_ecoHost_kegg.csv")
    for k,v in ecDict.iteritems():
        ec_number = k
        if len(v) == 1:## found only one KO id -> 1 EC number
            ko_id, gene = v[0]
            #print k,ko_id,gene
            caiScores = getAllGenesFromOrganisms(targetDir,ko_id)
            if caiScores != []:
                maxCai,oId,ORFid = caiScores[-1]
                orgName = orgNameDict[oId]
                rec =
[ec_number, 'KEGG',gene,maxCai,ORFid,oId,orgName,ko_id]
                printListToFile(outputFile,rec,"\t")

            else:
                print "Empty CAI not Found!!!!!!!!!!","\t",ec_number

        else:## found more than one KO id -> 1 EC number
            caiDict = {}
            for kdata in v:

```

```

                ko_id, gene = kdata
                #print k,ko_id,gene
                caiScores = getAllGenesFromOrganisms(targetDir,ko_id)
                caiDict[ko_id] = [caiScores,gene]
            allResults =
getCAIFromAllSubunit(caiDict)##(avg_cai,oId,caiOrf,"###".join(allKK),"###".jo
in(allGene))

            if allResults != []:

                result = allResults[-1]
                maxCai = result[0]
                oId = result[1]
                ORFids = result[2]
                koIDs = result[3]
                genes = result[4]
                orgName = orgNameDict[oId]

                rec =
[ec_number, 'KEGG',genes,maxCai,ORFids,oId,orgName,koIDs]
                printListToFile(outputFile,rec,"\t")
            else:
                print "Empty CAI not Found!!!!!!!!!!","\t",ec_number

        end = time.clock()
        print "Time elapsed = ", (end - start)/3600, "hrs"
    if __name__=='__main__':main()

```

A.3 Model simulations by MATLAB®

A.3.1 Main script “iJR904_maxTarget_r_glpk_par.m”

```
clear
clc
recycle('on');
curr_path= pwd;
cd(curr_path);
funct_folder = addpath(strcat(curr_path, '\function'));
start_time = clock;

%% clock returns a 6-element date vector containing the current date and
time in decimal form:
%% [year month day hour minute seconds]

%%%%%%%% In put files for simulation %%%%%%%%%
rxn_ext_file_nh3 = fullfile(curr_path, '\input\iJR904_reaction_ext.xls'); %%
external reaction ( substrate uptake and product secretion
rxn_c_nh3_file = fullfile(curr_path, '\input\iJR904_reaction_cyt.xls'); %%
cytosol reaction including Biomass equation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%% Case Basic condition %%%%%%%%%
% lb(v.glucose,1)= -20;
% ub(v.glucose,1) = -20;
%
% lb(v.atpm,1) = 7.6;
% ub(v.atpm,1) = 7.6;
%
% lb(v.o2,1)= -20;
% ub(v.o2,1) = 0;
%
% lb(v.nh3,1) = -100;
% ub(v.nh3,1) = 0;
%
% lb(v.so3,1) = -100;
% ub(v.so3,1) = 0;
%
% lb(v.pi,1) = -100;
```

```
% ub(v.pi,1) = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[wt ,v] = createLPMatrixBasicCon(rxn_c_nh3_file,rxn_ext_file_nh3);
save wt_data_basic_condition.mat wt;

% [wt ,v] = createLPMatrixMicroaerobic(rxn_c_nh3_file,rxn_ext_file_nh3);
% save wt_eco_microaerobic_condition.mat wt;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Write simulation flux of wild type to text file %%%%%%%%%
% fid = fopen('iJR904_wt_flux_microaerobic.txt','w');
% for aa =1:1:size(wt.reaction_id,1);
%     if iscell(wt.reaction_id{aa,1}) ==0 || iscell(wt.reaction_id{aa,2})==
0;
%         rxn_data = wt.reaction_id{aa,1};
%         rxn_id = wt.reaction_id{aa,2};
%         rxn_no = wt.reaction_id{aa,3};
%         rxn_flux = num2str(wt_flux(aa));
%     else
%         rxn_data = wt.reaction_id{aa,1}{:};
%         rxn_id = wt.reaction_id{aa,2}{:};
%         rxn_no = wt.reaction_id{aa,3};
%         rxn_flux = num2str(wt_flux(aa));
%     end
%     fprintf(fid,rxn_data());
%     fprintf(fid,'\t');
%     fprintf(fid,rxn_id());
%     fprintf(fid,'\t');
%     fprintf(fid,num2str(rxn_no));
%     fprintf(fid,'\t');
%     fprintf(fid,rxn_flux);
%     fprintf(fid,'\n');
%
% end
% fclose(fid);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% END %%%%%%%%%
```

```
totalIteration = 23;
matlabpool open 12
timeCals=zeros(totalIteration,3);
for iteration=1:totalIteration;
```



```

    if
exist(strcat(curr_path,'iteration_',num2str(iteration)), 'dir')==0; %% if no
Directory, make directory for data
        mkdir(curr_path,
strcat('iteration_',num2str(iteration)));
        end
        nonNativeMetPath =
strcat('C:\Users\sunisa\Desktop\sunisa\2012SEP_BKM_react\addReactions\assum
e_r\eco_add\iteration_',num2str(iteration));
        listAllFiles = dir(fullfile(nonNativeMetPath, '*.txt'));
        each_iteration_result =
fullfile(strcat('iJR904_maxTarget_producible_basicCon_r_',num2str(iteration
),'.csv'));
        load wt_data_basic_condition.mat;
tic;
        parfor fName=1:length(listAllFiles);
% parfor fName=1:10;
            targetFnamePath =
fullfile(nonNativeMetPath,listAllFiles(fName).name);
            curFname = listAllFiles(fName).name;
            rxn_data = textread(targetFnamePath, '%s');
            canRxnName = rxn_data(1);
            addRxns =
getAdditionalBKMrxn_mod(targetFnamePath,size(wt.reaction_matrix_ce,1));
            mutant_reaction_id = [wt.reaction_id;addRxns];

            canRxns = cellstr(rxn_data((2:end),1)); %% first line shows the
information of connectable metabolites (BKM-react ID)
            [mutant_rxns,mutant_metabolites,met_row] =
convertStoiAsNumber(canRxns,size(wt.metabolites,1),wt.metabolites);
            [reactionMatrixExpand,lb,ub,~] =
reactionMatrix(mutant_rxns,size(wt.reaction_matrix_ce,1)+1,wt.lb,wt.ub,wt.r
eaction_matrix_ce);

            %% add lower and upper of target reaction %%
            lb(size(reactionMatrixExpand,1),1) =0.0;
            ub(size(reactionMatrixExpand,1),1)=1000.0;
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %% Maximize biomass first %%%
            biomass_rxn = wt.biomass_rn;
            mutant_metabolites = unique(mutant_metabolites);
            mutant_lb = lb;
            mutant_ub = ub;

```

```

        targetReaction = size(reactionMatrixExpand,1); %% target reaction
of nonNative metabolite

        [LP_matrixExpand] =
makeLPMatrix_mod(reactionMatrixExpand,mutant_metabolites);
        [s,t] = size(LP_matrixExpand); %% s = no of metabolites, t = no of
reactions;
        b = zeros(s,1);
        c = zeros(1,t);
        c(1,v.biomass+1) = 1; %% determine objective function
        objType = -1 %% for gurobi solver -> 1 = minimize, -1 = maximize
obj function
        conType = repmat('S',1,s);
        varType = repmat('C',1,t);
        % [mt_flux,mt_fval,mt_exitflag,mt_output,mt_lambda] =
gurobi_mex(c',objType,LP_matrixExpand,b,conType,mutant_lb,mutant_ub,varType
);

        [mt_flux,mt_fval, mt_status, mt_extra] =glpk (c, LP_matrixExpand, b,
mutant_lb,mutant_ub, conType, varType, objType);
        mt_growth = num2str(mt_flux(v.biomass,1));
        disp(['Expand model maximize E. coli growth (assume r) = '
mt_growth 'solution status if 2 , unique == ' num2str(mt_status)])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %% Maximize Target production of mutant without fixing
growth %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        mt_lb = mutant_lb;
        mt_ub = mutant_ub;
        mt_lb(v.biomass+1,1) = 0;
        mt_ub(v.biomass+1,1) = mt_flux(v.biomass+1,1);

        v_carbon = mt_flux(v.glucose,1);
        [lpMatrixExpand] =
makeLPMatrix_mod(reactionMatrixExpand,mutant_metabolites);
        [m,n] = size(lpMatrixExpand);
        b_target = zeros(m,1);
        c_target = zeros(1,n);
        c_target(1,targetReaction) = 1; %% Target reaction
objType_target = -1; %% to maximize target reaction flux
conType_target = repmat('S',1,m);

```

```

        varType_target = repmat('C',1,n);
%
[mt2_flux,mt2_fval,mt2_exitflag,mt2_output,mt2_lambda] =
gurobi_mex(c_target',objType_target,lpMatrixExpand,b_target,conType_target,
mt_lb,mt_ub,varType_target);
[mt2_flux,mt2_fval, mt2_status, mt2_extra] =glpk
(c_target,lpMatrixExpand, b_target, mt_lb,mt_ub, conType_target,
varType_target, objType_target);

disp(['Expand model maximize target of E. coli growth (r)= '
num2str(mt2_flux(v.biomass)) 'At iteration = ' num2str(iteration)
'Solution status if 2, unique == ' num2str(mt2_status)])
disp(['Expand model maximize target of E. coli target (r)= '
num2str(mt2_flux(targetReaction)) 'At iteration = ' num2str(iteration)
'Solution station if 2, unique == ' num2str(mt2_status)])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% check target production higher than 1% of carbon source%%
if mt2_status ==5 && mt2_flux(targetReaction) > 0.00;
    t_flux = num2str(mt2_flux(targetReaction));
    t_growth = num2str(mt2_flux(v.biomass));

    wfile= fopen(each_iteration_result,'a');
    fprintf(wfile,'%s\t%s\t%s\t%s\n',curFname(1:end-
4),t_flux,t_growth,num2str(iteration));
    fclose(wfile);

end
outputFile =
strcat(curr_path,'\iteration_',num2str(iteration),'\',listAllFiles(fName).n
ame(1:end-4),'_Flux.txt');
writeOutputFluxToFile_par(outputFile,mutant_reaction_id,mt2_flux);

end

timeCals(iteration,1) = iteration;
timeCals(iteration,2) = length(listAllFiles);
% timeCals(iteration,2) = 10;
timeCals(iteration,3) = toc;

fTime = fopen('eco_maxTarget_r_glpk_time.txt','a');

fprintf(fTime,'%s\t%s\t%s\n',num2str(iteration),num2str(length(listAllFile
s)),num2str(toc));

```

```

fclose(fTime);

end
matlabpool close
xlswrite('eco_maxTarget_r_glpk.xls',timeCals);
disp('Finised calculation !!!! E. coli assume r BY GLPK solver ')

%% clock ==> [year month day hour minute seconds] %%
%% start_time(1,1) = year
%% start_time (1,2) = month
%% start_time (1,3) = day
%% start_time (1,4 ) = hour
%% start_time (1,5) = minute
%% start_time(1,6) = seconds

end_time = clock;
s_year = start_time(1,1);
s_month = start_time(1,2);
s_day = start_time(1,3);
s_hour = start_time(1,4);
s_min = start_time(1,5);

e_year = end_time(1,1);
e_month = end_time(1,2);
e_day = end_time(1,3);
e_hour = end_time(1,4);
e_min = end_time(1,5);

disp(['starting time Year Month Day Hour Minute = ' num2str(s_year)
num2str(s_month) num2str(s_day) num2str(s_hour) num2str(s_min)])
disp(['End time Year Month Day Hour Minute = ' num2str(e_year)
num2str(e_month) num2str(e_day) num2str(e_hour) num2str(e_min)])

```

A.3.2 “createLPMatrixBasicCon.m”

```
function [wt,v] =createLPMatrixBasicCon(rxn_c_nh3_file,rxn_ext_file_nh3)

[reaction_cytosol_data,reaction.cytosol_id] = getReactions(rxn_c_nh3_file);

[reaction_c.eq,reaction_c.metabolite,m_row]=convertStoiAsNumber(reaction_cy
tosol_data,0,[]);
[reaction_matrix_cytosol, lower_bound, upper_bound, ~] =
reactionMatrix(reaction_c.eq,1,[],[],{ });

[~,rxn_extracellular] = xlsread(rxn_ext_file_nh3);

[reaction_matrix_withEXT,reaction_ext_id,lb,ub] =
addExtReaction(rxn_extracellular,reaction_matrix_cytosol,lower_bound,upper_
bound);
met_ext = reaction_ext_id(1:end,1);
wt.cytosolRxns = reaction.cytosol_id;
wt.reaction_matrix_ce = reaction_matrix_withEXT;
wt.metabolite_ext = met_ext;
wt.reaction_id =[reaction.cytosol_id;reaction_ext_id];
wt.metabolite_c = reaction_c.metabolite;

wt.metabolites = unique([ reaction_c.metabolite;met_ext]);

[~,cc_rns] = xlsread(rxn_ext_file_nh3);
[~,v]=keepTransportRn(wt.reaction_id,cc_rns);

[LP_matrix_original] =
makeLPMatrix_mod(reaction_matrix_withEXT,wt.metabolites);

%%%%%% Case Basic condition %%%%%%%%%%

lb(v.glucose,1)= -20;
ub(v.glucose,1) = -20;

lb(v.o2,1)= -20;
ub(v.o2,1) = 0;

lb(v.nh3,1) = -20;
```

```
ub(v.nh3,1) = 0;

lb(v.so3,1) = -20;
ub(v.so3,1) = 0;

lb(v.pi,1) = -20;
ub(v.pi,1) = 0;

%%%%%%%%%% END%%%%%%%%%%

wt.lpMatrix = LP_matrix_original;
wt.biomass_rn =v.biomass;
wt.lb = lb;
wt.ub = ub;

end
```

A.3.3 “getReactions.m”

```
function [rn_eq,rn_tag]= getReactions(target_file)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% To collect all strings of metabolic reaction
%% and keep all strings in one column
%% outputs are rn_eq(n,1) and rn_tag (keep reaction ID)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% First to keep reaction tag in which to identify what target reaction
%% is?
%% Reaction data was read and keep in reaction_set variable:
%% reaction_set(1,1) = reaction_id
%% reaction_set(1,2) = reaction_palisson_id
%% reaction_set(1,3) = reaction_kegg_id
[~,reaction_set] = xlsread(target_file);
rxns =cell(size(reaction_set),3);
for v=1:size(reaction_set,1);
    rxns{v,1}= reaction_set(v,3);
    rxns{v,2} = reaction_set(v,1);
    rxns{v,3} = v;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%Second to keep metabolic information by split text data into cellstr data
%%by using space as delimiter.
%% Ex:
%%
{'FADH2_TCA[c]'; '+'; 'VITAMIN_K_{2}[c]'; '+'; '2.0'; 'C00080[e]'; '=>'; 'FAD_TCA[
c]'; '+'; 'REDUCED-MENAQUINONE[c]'; '+'; '2.0'; 'C00080[c]'; }
```

```
%%The vector delimiter contains valid delimiter characters. Any
leading delimiters are ignored.
[str,remain] = strtok(remain);
if strcmp(str,{' '});
    kk=2;%% update kk cause no more string in this reaction
end
position = position+1;
result{position,1} = str; %% update position for collecting str
found in each reaction

end
elseif k>1;%% not first reaction
    remain = rxns{k,1};
    kkk = 0;
    while kkk ==0;
        [str,remain] = strtok(remain);
        if strcmp(str,{' '});
            kkk = 1;%% update kkk cause no more string in this reaction
        else
            result{position,1} = str;
            position = position+1;
        end
    end

end % if k==1
end % for k=1:size(rxns,1)
rn_eq = result;
rn_tag = rxns;
end
```

A.3.4 “convertStoiAsNumber.m”

```
function [reaction metabolite met_rows] =
convertStoiAsNumber(rxn_cytosol,m_rows, metabolite_list)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% From getReactions function,
%% all data is in string type as well as
%% stoichiometry coefficient.
%% To covert the stoichiometry coefficient to number (n,1)
%% and collect metabolite ID
%% outputs are reaction array (n,1) and metabolites (m,1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% To convert stoichiometry as a number
for i=1:size(rxn_cytosol,1); % size of rxn_cytosol equal to a member of
array

    if iscell(rxn_cytosol{i,1});%% convert cell to char for comparing with
string
%       rxn_cytosol{i,1} = char(rxn_cytosol{i,1}{:});
temp = char(rxn_cytosol{i,1}{:});

        if str2num(temp) ~= 0;%% if temp is numeric:
            temp = str2num(temp);
            rxn_cytosol{i,1} = temp;
        else
            pat = '\w+\{+[a-z]+\}'; %% find pattern of compartment such
as[c], [m] ..
            if regexp(temp, pat) ~= 0; %% found metabolites from the
compartment
                m_rows = m_rows+1; %% set position of metabolite in array
                metabolite_list{m_rows,1} = temp;
                rxn_cytosol{i,1} = temp;
            else
                temp = temp;
                rxn_cytosol{i,1} = temp;
            end
        end
    end
else
    temp = rxn_cytosol{i,1};
end
```

```
        if str2num(temp) ~= 0;%% if temp is numeric:
            temp = str2num(temp);
            rxn_cytosol{i,1} = temp;
        else
            pat = '\w+\{+[a-z]+\}'; %% find pattern of compartment such
as[c], [m] ..
            if regexp(temp, pat) ~= 0; %% found metabolites from the
compartment
                m_rows = m_rows+1; %% set position of metabolite in array
                metabolite_list{m_rows,1} = temp;
                rxn_cytosol{i,1} = temp;
            else
                temp = temp;
                rxn_cytosol{i,1} = temp;
            end
        end
    end

end
metabolite = metabolite_list;
reaction = rxn_cytosol;
met_rows = m_rows;
```

A.3.5 “reactionMatrix.m”

```
function [matrix, lb, ub, row] =
reactionMatrix(rxn_cytosol,rows,lower_bound,upper_bound,reaction_matrix)

% create reaction matrix
% input reaction in cytosol
% retrun reaction matrix, lowerbound, upperbound and row of reaction

% reaction_matrix = {};
% rows = 1;
% lower_bound = [];
% upper_bound = [];
columns = 1;
% Making reaction matrix %%%

for i=1:size(rxn_cytosol,1)-1;% minus one because last metabolite
determine at line 61
    reaction_matrix{rows,columns} = rxn_cytosol{i};
    if isnumeric(rxn_cytosol{i}) ==1;
        rows = rows; % found stoichiometry so row number not increase

        elseif strcmp(rxn_cytosol{i},'+') == 1 | strcmp(rxn_cytosol{i},'=>')
== 1 | strcmp(rxn_cytosol{i},'<=>') == 1;
            rows = rows;

            if strcmp(rxn_cytosol{i},'=>') == 1;

                lower_bound(rows,1)= 0.00;
                upper_bound(rows,1) = 1000.0;

            elseif strcmp(rxn_cytosol{i},'<=>') == 1;

                lower_bound(rows,1)=(-1)*(1000.0);
                upper_bound(rows,1) = 1000.0;

            end

        elseif isnumeric(rxn_cytosol{i}) == 0 & strcmp(rxn_cytosol{i},'+') ==
0 & strcmp(rxn_cytosol{i},'=>') == 0 ...
            & strcmp(rxn_cytosol{i},'<=>')== 0;
            %%% final metabolite name (cpd_id) from previous reaction %%%
```

```
        if ~isnumeric(rxn_cytosol{i+1}) & ~strcmp(rxn_cytosol{i+1},'+') &
~strcmp(rxn_cytosol{i+1},'=>') & ~strcmp(rxn_cytosol{i+1},'<=>');
            %% Next array should be metabolite (next reaction) so increase
            %% row (number of reaction increase)
            rows = rows+1;
            columns = 0;
        elseif isnumeric(rxn_cytosol{i+1}) == 1 %% only stoichiometry
before the first metabolite of this reaction
            %% next array must be metabolite for starting reaction
            if ~isnumeric(rxn_cytosol{i+2}) & ~strcmp(rxn_cytosol{i+2},'+')
& ~strcmp(rxn_cytosol{i+2},'=>') & ~strcmp(rxn_cytosol{i+2},'<=>');
                rows = rows+1;
                columns = 0;

            end
        end
        columns = columns+1; %% if not metabolite have to extend column for
equal size of all reaction

    end

    reaction_matrix {rows, columns} = rxn_cytosol{size(rxn_cytosol,1)}; %%
determine the last metabolite of reaction
    rows = rows+1; % adding one row for starting next reaction from external
reaction
    matrix = reaction_matrix;
    lb = lower_bound;
    ub = upper_bound;
    row = rows;
```

A.3.6 “makeLPMatrix_mod.m”

```

function [LP_matrix] =
makeLPMatrix_mod(reaction_matrix_withEXT,Allmetabolites)

%%% Making LP matrix %%%%

[m,n] = size(reaction_matrix_withEXT);
LP_matrix = sparse(length(Allmetabolites),m);

%%% construct stoi matrix %%%%
nonMetsList = {'<=', '=>', '<=>'};
for i =1:m
    oneRxn = reaction_matrix_withEXT(i,1:end);
    for sign=1:3
        pos = find(strcmp(nonMetsList(sign),oneRxn));
        if (~isempty(pos))
            sign_pos = pos;
        end
    end
    reactant_mets = reaction_matrix_withEXT(i,1:sign_pos-1);
    product_mets = reaction_matrix_withEXT(i,sign_pos+1:end);

    for k=1:length(reactant_mets);
        item = reactant_mets{1,k};
        m_pos = find(strcmpi(item,Allmetabolites));

        if (~isempty(m_pos)) && k==1; %% found start metabolite %%

            LP_matrix(m_pos,i) = -1;

        elseif (~isempty(m_pos)) && k > 1; %% found metabolite %%

            if isempty(reactant_mets{1,k-1})||(strcmp(reactant_mets{1,k-1},'+'))==1;
                LP_matrix(m_pos,i) = -1;
            elseif isnumeric(reactant_mets{1,k-1});
                LP_matrix(m_pos,i) = (-1)*(reactant_mets{1,k-1});
            end
        end
    end

    end

end

end

for k=1:length(product_mets);
    item = product_mets{1,k};
    m_pos = find(strcmpi(item,Allmetabolites));

    if (~isempty(m_pos)) && k==1; %% first of product
        LP_matrix(m_pos,i) = 1;

    elseif (~isempty(m_pos)) && k >1;

        if isempty(product_mets{1,k-1})||(strcmp(product_mets{1,k-1},'+'))==1;
            LP_matrix(m_pos,i) = 1;
        elseif isnumeric(product_mets{1,k-1});
            LP_matrix(m_pos,i) = (1)*(product_mets{1,k-1});
        end
    end

end

end

```

A.3.7 “addExtReaction.m”

```
function [reaction_matrix_withEXT, reaction_ext_id, lower_bound, upper_bound]
= addExtReaction(rxn_ext_item, reaction_matrix, lb, ub)

%% add extracellular reaction %%
%% rxn_ext_item(1,1:end) = {'[e]12ppd-S', 'C02917', '=>'};

reaction_ext_id = cell(size(rxn_ext_item,1),2);
rows = size(reaction_matrix,1);
rows = rows+1;
columns = 1;

for i=1:size(rxn_ext_item,1);

    reaction_ext_id{i,1} = strcat(rxn_ext_item{i,2}, '[e]'); %% metabolite
    reaction_ext_id{i,2} = rxn_ext_item{i,1}; %% reaction id
    reaction_ext_id{i,3} = rows; %% no_reaction in reaction_matrix

    reaction_matrix{rows, columns} = strcat(rxn_ext_item{i,2}, '[e]'); %%
    external metabolite in KEGG ID

    if strcmp(rxn_ext_item{i,3}, '=>')==1;
        lb(rows,1) = 0.0;
        ub(rows,1) = 1000.0;
        columns = columns+1;
        reaction_matrix{rows, columns} = rxn_ext_item{i,3};
    elseif strcmp(rxn_ext_item{i,3}, '<=>')==1;
        lb(rows,1) = -1000.0;
        ub(rows,1) = 1000.0;

        columns = columns+1;
        reaction_matrix{rows, columns} = rxn_ext_item{i,3};
    end
    rows = rows+1;
    columns=1;
lower_bound = lb;
upper_bound = ub;
reaction_matrix_withEXT = reaction_matrix;

end
```

A.3.8 “getAdditionalBKMrxn_mod.m”

```
function no_add_rxns =
getAdditionalBKMrxn_mod(target_fName_path, original_reaction_no)
f_target = fopen(target_fName_path, 'r');
tLine = fgetl(f_target);
fLine = tLine;
findRxnPos = regexpi(fLine, 'R[0-9][0-9][0-9][0-9][0-9]');
rxnsList = cell(size(findRxnPos,2),1);

for item=1:size(findRxnPos,2);
    rn_id = fLine(findRxnPos(item):findRxnPos(item)+5);
    rxnsList{item} = rn_id;
end

rxnsList = unique(rxnsList);
rxnsList = [rxnsList; 'addRxn1'; 'addRxn2']; %% add transporter reactions (by
diffusion)
no_add_rxns = cell(length(rxnsList),3);

pos = 0;
while pos < length(no_add_rxns);
    pos = pos+1;
    tLine = fgetl(f_target);
    no_add_rxns{pos,1} = tLine;
    no_add_rxns{pos,2} = rxnsList{pos};
    no_add_rxns{pos,3} = original_reaction_no+pos;

end
fclose(f_target);
```


A.3.9 “keepTransportRn.m”

```
function [central_reaction,v ]= keepTransportRn(reaction_id,t_reaction)
central_reaction = cell(size(t_reaction,1),4);
for vv=1:size(t_reaction,1);
    target_rnID = t_reaction(vv,1);

    for k=1:size(reaction_id,1);
        x= reaction_id(k,2);
        r_id = x{:};

        if strcmpi(r_id,target_rnID) ==1;
            central_reaction{vv,1} =reaction_id{k,3};
            central_reaction{vv,2} = t_reaction(vv,1);%% rn_id
            central_reaction{vv,3} = t_reaction(vv,2);%% ec_id
            central_reaction{vv,4} = t_reaction(vv,3);%% pathway name

        end

        if strcmp(r_id,'BIOMASS_TRANS')==1;
            v.biomass =reaction_id{k,3};

        elseif strcmp(r_id, '[e]etoh')==1;
            v.eth = reaction_id{k,3};

        elseif strcmp(r_id,'[e]glc-D')==1;
            v.glucose = reaction_id{k,3};

        elseif strcmp(r_id,'[e]o2')==1;
            v.o2 =reaction_id{k,3};

        elseif strcmp(r_id,'[e]ac')==1;
            v.acetate =reaction_id{k,3};

        elseif strcmp(r_id,'[e]lac-D')==1;
            v.lactate =reaction_id{k,3};

        elseif strcmp(r_id,'[e]succ')==1;
            v.succ =reaction_id{k,3};

        elseif strcmpi(r_id,'[e]co2')==1;
```

```
            v.co2=reaction_id{k,3};

        elseif strcmp(r_id,'[e]nh3')==1;
            v.nh3=reaction_id{k,3};

        elseif strcmp(r_id,'[e]pi')==1;
            v.pi =reaction_id{k,3};

        elseif strcmp(r_id,'[e]so4')==1;
            v.so3 =reaction_id{k,3};

        elseif strcmp(r_id,'[e]h2o')==1;
            v.h2o =reaction_id{k,3};
        elseif strcmp(r_id, 'ATPM') == 1;
            v.atpm = reaction_id{k,3};

        end
    end

end

end
```

A.3.10 “writeOutputFluxToFile.m”

```
function
writeOutputFluxToFile_par(fileName,mutant_reaction_id,mutant_maxTargetFlux)

f_flux = fopen(fileName,'w');
flux = mutant_maxTargetFlux;
parfor aa =1:1:size(mutant_reaction_id,1);
    if iscell(mutant_reaction_id{aa,1}) ==0 ||
iscell(mutant_reaction_id{aa,2})== 0;
        rxn_data = mutant_reaction_id{aa,1};
        rxn_id = mutant_reaction_id{aa,2};
        rxn_no = mutant_reaction_id{aa,3};
        rxn_flux = num2str(flux(aa));
    else
        rxn_data = mutant_reaction_id{aa,1}{:};
        rxn_id = mutant_reaction_id{aa,2}{:};
        rxn_no = mutant_reaction_id{aa,3};
        rxn_flux = num2str(flux(aa));
    end
    fprintf(f_flux,rxn_data());
    fprintf(f_flux,'\t');
    fprintf(f_flux,rxn_id());
    fprintf(f_flux,'\t');
    fprintf(f_flux,num2str(rxn_no));
    fprintf(f_flux,'\t');
    fprintf(f_flux,rxn_flux);
    fprintf(f_flux,'\n');
end
fclose(f_flux);
```

Appendix B

Table B.1 Values of relative adaptiveness (w) of codon generated from highly expressed genes of *B. subtilis*, *E. coli*, and *S. cerevisiae*.

		<i>B. subtilis</i>	<i>E. coli</i>	<i>S. cerevisiae</i>			<i>B. subtilis</i>	<i>E. coli</i>	<i>S. cerevisiae</i>
ala	GCA	0.926	0.715	0.094	leu	CTA	0.120	0.007	0.149
	GCC	0.263	0.504	0.428		CTC	0.142	0.091	0.015
	GCG	0.700	1.000	0.020		CTG	0.353	1.000	0.043
	GCT	1.000	0.815	1.000		CTT	1.000	0.060	0.066
arg	AGA	0.304	0.005	1.000		TTA	0.508	0.034	0.310
	AGG	0.004	0.002	0.052		TTG	0.263	0.051	1.000
	CGA	0.059	0.009	0.005	lys	AAA	1.000	1.000	0.328
	CGC	0.783	0.529	0.015		AAG	0.205	0.244	1.000
	CGG	0.023	0.004	0.004	met	ATG	1.000	1.000	1.000
	CGT	1.000	1.000	0.213					
asn	AAC	1.000	1.000	1.000	phe	TTC	1.000	1.000	1.000
	AAT	0.480	0.189	0.309		TTT	0.726	0.400	0.402
asp	GAC	0.759	1.000	1.000	pro	CCA	0.622	0.179	1.000
	GAT	1.000	0.844	0.896		CCC	0.022	0.016	0.034
cys	TGC	1.000	1.000	0.242		CCG	0.829	1.000	0.014
	TGT	0.783	0.550	1.000		CCT	1.000	0.122	0.212
gln	CAA	1.000	0.210	1.000	ser	AGC	0.567	0.814	0.120
	CAG	0.461	1.000	0.091		AGT	0.187	0.123	0.128
glu	GAA	1.000	1.000	1.000		TCA	0.645	0.146	0.165
	GAG	0.288	0.290	0.122		TCC	0.191	0.873	0.634
gly	GGA	0.820	0.033	0.042		TCG	0.046	0.218	0.044
	GGC	1.000	0.869	0.095		TCT	1.000	1.000	1.000
	GGG	0.128	0.071	0.022	stop	TAA	1.000	1.000	1.000
	GGT	0.840	1.000	1.000		TAG	0.057	0.009	0.173
his	CAC	0.966	1.000	1.000		TGA	0.028	0.135	0.204
	CAT	1.000	0.382	0.526	thr	ACA	1.000	0.067	0.167
ile	ATA	0.031	0.004	0.076		ACC	0.078	1.000	0.840
	ATC	1.000	1.000	0.959		ACG	0.322	0.195	0.035
	ATT	0.895	0.487	1.000		ACT	0.552	0.488	1.000
trp					trp	TGG	1.000	1.000	1.000
tyr					tyr	TAC	1.000	1.000	1.000
						TAT	0.843	0.530	0.274
val					val	GTA	0.742	0.494	0.064
						GTC	0.399	0.324	0.712
						GTG	0.410	0.721	0.081
						GTT	1.000	1.000	1.000

List of publications

Chatsurachai, S., Furusawa, C., and Shimizu, H. (2012). An *in silico* platform for the design of heterologous pathways in nonnative metabolite production. *BMC Bioinformatics*, 13, 93.

Chatsurachai, S., Furusawa, C., and Shimizu, H. (2013). ArtPathDesign – Rational heterologous pathway design system for the production of nonnative metabolites. *J Biosci Bioeng* (in press)

Acknowledgements

This thesis would not be possible without the contribution of many people. I would like to express my deepest gratitude to Prof. Dr. Hiroshi Shimizu, Metabolic Engineering, Department of Bioinformatic Engineering, Graduate School of Information Science and Technology, Osaka University, who has given me an opportunity to study in his laboratory and for endless support, invaluable guidance, patience, and encouragement for my research.

I would like to express my sincere appreciation to Prof. Dr. Hisao Ohtake and Prof. Dr. Toshiya Muranaka for their helpful comments and suggestions to improve my thesis. I would like to express my sincere gratitude to Prof. Dr. Chikara Furusawa for the immeasurable amount of support, endless patience, and careful guidance he has provided throughout this study.

I would like to thank to Assoc. Prof. Dr. Fumio Matsuda for his helpful advices, comments, and encouragements.

I am especially grateful to all members in Shimizu's laboratory for their help throughout my study.

I would like to gratefully acknowledge the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan for the Japanese government (Monbukagakusho) scholarship.

I would like to thank my best friend Aiyada Aroonsri for her helpful advices and support for all these years.

I would like to thank all my friends for their support throughout these years in Japan. Without them, my life in Japan is not go as nice as this.

I am especially grateful to my parents for their support, love and continuous encouragements throughout many years.

Sunisa Chatsurachai