Optimization strategies for metabolic networks

Alexandre Domingues, J. M. Lemos, Susana Vinga

Abstract—The progressive availability of models and data for metabolic networks poses new challenges in what concerns optimization. Due to the high level of complexity and uncertainty associated to these networks the suggested models often lack detail and liability, required to determine the proper optimization strategies. A possible approach to overcome this limitation is the combination of both kinetic and stoichiometric models. In this paper three approaches, with different levels of complexity, are presented and their results compared using a prototype network.

I. INTRODUCTION

With current metabolic engineering processes it is possible to manipulate metabolic networks to improve desired characteristics. These manipulations may lead to the maximization of the normal product yield or even redirect the production to a flux that was residual or non-significant in the original network.

The high level of uncertainty of metabolic network models makes it extremely difficult to determine what are the required manipulations needed to attain a certain objective. An heuristic approach to such problems does not allow to explore the maximum potential of metabolic engineering.

When optimizing a metabolic network for a given objective two distinct problems must be addressed. The first is to determine which branch or branches must be manipulated. The second is to determine what type of manipulations must be done. Strategies such as OptKnock[4] address the first problem, in this report a strategy for the second problem is described.

A common optimization problem is the maximization of the final concentration of a metabolite whose formation competes with the natural objective of the cell (e.g. maximization of biomass). In this work, a prototype network with such behavior is taken as example and the optimization problem is solved.

A. State of the art

Although Metabolic Engineering [10] has developed very powerful approaches to optimize biotechnological processes, the systematic use of model base and optimal control methods is still reduced and poses many open issues. Interesting examples are provided, some at the genome level, by [11] [12] [3] [13]. In [3] the use of a bi-level optimization method, including a linear programming problem in the inner level and a nonlinear optimization problem in the outer level, presents the interesting feature of not requiring the full model knowledge. This optimization method is used on an In silico model of E.Coli and tested In vivo with promising results. The work in [5] and [6] focus on techniques to determine dynamic distributions of fluxes on metabolic network models where not all the kinetics are known.

B. Paper contributions and organizations

The major paper contributions consist in the development of a case study on metabolic network optimization where three different methods are compared. Two of these methods assume complete knowledge of the dynamic equations of a network model. While one relies on an Optimal Control approximation, the other makes a steady-state optimization using Geometric Programming. These methods provide a base line performance with which the results obtained by other methods may be compared. As such, the third approach assumes only a partial knowledge of the network kinetic model and relies on a bi-level optimization. Furthermore, using Pontryagin’s Maximum Principle, it is shown that, for the class of problems considered, the manipulated variable may only assume values at the extremes of the optimization interval. The paper is organized as follows: After the Introduction (Section I) in which the problem is introduced and motivated and the state of the art revised, the problem is formulated in Section II, where the prototype network model is described. Section III formulates the optimization methods and discusses properties of the optimal solution and Section IV presents numerical results. Finally, conclusions are drawn in Section V.

II. PROBLEM FORMULATION

A. Prototype network model

The optimization strategies were tested on a prototype network. This network is an adaptation of a previously suggested network [1] and is described by the set of ordinary differential equations shown in (1).

\[
\begin{align*}
\frac{du_1}{dt} &= k - v_1 \\
\frac{dx_2}{dt} &= v_1 - v_2 - v_3 \\
\frac{du_3}{dt} &= v_2(1 - f) \\
\frac{dx_4}{dt} &= v_3(f) - v_4 \\
\frac{du_5}{dt} &= v_4
\end{align*}
\]
is biased towards the branch of \( v_3 \) the production of \( u_5 \) will be affected by the low concentration of \( u_3 \) (since there is a forward feedback). Thus, there is an optimal profile for \( f(t) \) to maximize the concentration of \( u_5 \) at the final time \( t_{final} \).

### B. The optimization problem

The optimization problem consists in selecting \( f(t) \) for \( t \) in the interval \([0, \ldots, t_{final}]\) such that:

\[
J(f) = u_5(t_{final})
\]

is maximized.

### III. OPTIMIZATION METHODS

#### A. Flux Balance analysis

The difficulties that arise when determining the detailed kinetic information on metabolic networks led to the formulation of new approaches to network modeling. One of them, Flux Balance Analysis (FBA), has proven useful in the study of metabolic systems [7] [5] [8] and is part of the optimization process of the current study. The first step on FBA is the reconstruction of the metabolic network, such as in Fig. 1. Mass balance equations are written around every metabolite (4), and known constraints (such as lower and upper bounds for fluxes) are included (5).

\[
\frac{dX}{dt} = S.v
\]

Where \( \frac{dX}{dt} \) is a vector with the instant variation of each metabolite, \( S \) is a matrix containing the stoichiometry of the catabolic reactions, \( v \) a vector of the “\( n \)” metabolic reactions rates, \( \alpha \) and \( \beta \) are the lower and upper constraints for each flux. If it is assumed that the system has achieved steady-state, (4) becomes \( S.v = 0 \), which is typically an underdetermined equation since there are more fluxes than metabolites. A particular solution can be found by solving a problem of linear programming (LP) with the proper objective function. In optimal environmental conditions, with enough substrate, it is valid to assume that the cellular objective is the maximization of biomass [8], thus the objective function of the LP can be a flux or a function of fluxes known to be related to growth precursors. The FBA framework has been extended [5] [6] to incorporate the dynamics of the networks. Dynamic Flux Balance Analysis (dFBA) can predict the reprogramming of a metabolic network and model the dynamics of certain metabolites over time, this is done by solving the steady-state problem at several time instants and integrating the known fluxes during each time interval.

#### TABLE I

| Param. | Value | Param. | Value |
|--------|-------|--------|-------|
| \( \alpha_2 \) | 8 | \( h_{11} \) | 0.5 |
| \( \alpha_3 \) | 4.0556 | \( h_{22} \) | 1.4224 |
| \( \alpha_4 \) | 1.8397 | \( h_{33} \) | 0.6109 |
| \( \alpha_5 \) | 4.0556 | \( h_{44} \) | 0.5829 |
| \( \beta_1 \) | 1 | \( g_{21} \) | 0.5 |
| \( \beta_2 \) | 5.1179 | \( g_{22} \) | 0.4171 |
| \( \beta_4 \) | 4.0556 | \( g_{24} \) | 2.8274 |
| \( k \) | 0.5 | \( g_{33} \) | 1.4646 |
| | | \( g_{34} \) | 0.5 |

Where \( du_i, i = 1, 3, 5 \) and \( x_i, i = 2, 4 \) are metabolite concentrations at the network nodes, \( v_i, i = 1, \ldots, 4 \) are fluxes associated to branches and \( \alpha \) is a constant parameter that represents the yield of \( u_1 \). In the equations, \( \alpha_i, i = 1, \ldots, 5 \) are constant parameters. They were adapted from the initial model [1] and adjusted to obtain the desired response. Table I shows the list of parameters. To distinguish between metabolites concentrations and inputs/outputs different letters were used, thus, \( x \) represents the concentration of a metabolite and \( u \) an input/output.

A graphical representation of the network is shown in Fig. 1.

![Prototype network](image)

Fig. 1. Prototype network: The maximization of the final value of \( u_5 \) depends on the profile of the function \( f(t) \).

Assuming that \( u_3 \) represents a precursor of the cellular objective (such as growth) and \( u_5 \) the desired product, if \( f(t) \) is biased towards the branch of \( v_2 \) this yields the formation of \( u_3 \) but little or no production of \( u_5 \). If \( f(t) \) is biased towards the branch of \( v_3 \) the production of \( u_5 \) will be affected by the low concentration of \( u_3 \) (since there is a forward feedback).

\[
\frac{du_1}{dt} = k - \beta_1 u_1^{h_{11}}
\]
\[
\frac{dx_2}{dt} = \alpha_2 u_1^{g_{21}} - \beta_2 x_2^{h_{22}}
\]
\[
\frac{dx_3}{dt} = \alpha_3 (1 - f) x_2^{g_{32}}
\]
\[
\frac{dx_4}{dt} = \alpha_4 (f) u_3^{g_{43}} x_2^{g_{32}} - \beta_4 x_4^{h_{44}}
\]
\[
\frac{dx_5}{dt} = \alpha_5 x_4^{g_{54}}
\]
B. Geometric Programming

Geometric Programming is a powerful mathematical optimization tool that can be used in problems where the objective and constraint functions have a special form [16]. GP is of particular interest because it can solve large scale problems with extreme efficiency and reliability [15], it has been shown [14] that a problem formulated in S-Systems form can be solved with GP after a minimum adaptation.

Let \( x = (x_1, \ldots, x_n) \) be a vector of \( n \) real positive variables \( x_1, \ldots, x_n \). A function \( f(x) \) with the form

\[
f(x) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}
\]

is called a monomial function [16]. Where \( c > 0 \) and \( a_i \in \mathbb{R} \). A sum of one or more monomials is called a posynomial function[16] and any monomial is also a posynomial. The standard Geometric Programming problem is formulated as:

\[
\begin{align*}
\text{minimize} & \quad f_0(x) \\
\text{subject to} & \quad f_i(x) \leq 1; \quad i = 1, \ldots, m \\
& \quad g_i(x) = 1; \quad i = 1, \ldots, p,
\end{align*}
\]

Where \( f_i \) and \( f_0 \) are posynomial functions, \( g_i \) are monomials, and \( x_i \) are the variables to be optimized. Given that monomials are closed under multiplication and division (if \( f \) and \( g \) are both monomials then so are \( f \times g \) and \( f \div g \)) [16], transforming an S-Systems model (in steady state) to be used in a GP problem constraints is straightforward:

\[
\begin{align*}
\frac{dx_n}{dt} &= \alpha_n x_1^{\beta_{n,1}} x_2^{\beta_{n,2}} \cdots x_n^{\beta_{n,n}} - \beta_n x_1^{\alpha_{n,1}} x_2^{\alpha_{n,2}} \cdots x_n^{\alpha_{n,n}} \\
0 &= \alpha_n x_1^{\beta_{n,1}} x_2^{\beta_{n,2}} \cdots x_n^{\beta_{n,n}} - \beta_n x_1^{\alpha_{n,1}} x_2^{\alpha_{n,2}} \cdots x_n^{\alpha_{n,n}} \\
\alpha_n x_1^{\beta_{n,1}} x_2^{\beta_{n,2}} \cdots x_n^{\beta_{n,n}} &= \beta_n x_1^{\alpha_{n,1}} x_2^{\alpha_{n,2}} \cdots x_n^{\alpha_{n,n}}
\end{align*}
\]

Starting from the standard form of S-Systems (7) and assuming Steady-State (8) the expression is re-arranged (10) to the form of a GP problem constraint (6).

C. Pontryagin’s Maximum Principle

Let \( x_i \) be the set of state variables of a dynamical system with control inputs \( u_j \) such that

\[
\dot{x} = f_i(x, u_j), \quad x_i(0) = x_0, \quad u_i(t) \in U, \quad t \in [0, T]
\]

Where \( U \) is the set of valid control inputs and \( T \) is the final time. The control functions \( u_j \) must be chosen to maximize the functional \( J \), defined by

\[
J(u) = \psi(x_i(T)) + \int_0^T L(x_i(t), u_j(t)) \, dt
\]

The Hamiltonian is defined as:

\[
H(\lambda(t), x(t), u(t), t) = \lambda^T f(x(t), u(t)) + L(x(t), u(t))
\]

If we are only interested on the final state, \( L(x(t), u(t)) = 0 \). The optimum value of the control variables \( u_i \) at each time \( t \) are the ones that maximize \( H \). The optimal value for \( u_j(t) \) is such that:

\[
\frac{\partial H}{\partial u_j(t)} = 0
\]

Which means that the optimal value for \( u_j(t) \) is on the extreme values of \( U \).

D. Optimization

The model described in the previous section is optimized to obtain a maximum yield of \( u_5 \) at the end of the run-time \( (t_{final}) \). The only manipulated variable is \( f(t) \), that allows to control the fluxes \( v_3 \) and \( v_2 \). It is important to note that, according to the Pontryagin maximum principle, the control function \( f(t) \) can only assume the value of 0 or 1. Two different methods are used to determine the profile of \( f(t) \) that maximizes the product yield. On the first method the function \( f(t) \) is forced to have the profile shown in (11), this imposes that the branch \( v_2 \) is active in the beginning \( (f(t) = 0) \), building up biomass , switching then to the branch \( v_3 \) \( (f(t) = 1) \) and activating the production of \( u_5 \). The instant where the switch occurs will be called time of regulation \( (t_{reg}) \) from now on. This simulation tests all the possible values of \( t_{reg} \) and returns the function:

\[
J(t_{reg}) = u_5(t_{final})
\]

The value of \( t_{reg} \) that results on a maximum product yield is then determined.

\[
f(t) = \begin{cases} 0 & \text{if } t \leq t_{reg} \\ 1 & \text{if } t > t_{reg} \end{cases}, \quad t = 0, \ldots, t_{final}
\]

On the second method the function \( f(t) \) is divided in several time intervals and an optimization algorithm is run to determine the optimal value for \( f(t) \) at each time interval. Increasing the number of intervals results in an increased time resolution for \( f(t) \) but also increases the computation time. For \( n \) time intervals the optimization algorithm outputs:

\[
f(t) = f_i,
\]

for \( t = \frac{t_{final}}{n}, \ldots, \frac{t_{final}}{n} \), \( i = 1, \ldots, n \)

The only constraint applied to this optimization is \( 0 \leq f(t) \leq 1 \).

All simulations assume \( t_{final} = 30s \). The software was implemented using Matlab, linear programming problems were solved using the function linprog and non-linear problems using the function fmincon. For Geometric Programming problems, functions from the gplab[9] package were used. The simulations were run on a laptop with a 1.6GHz processor and 512mB of Ram.

1) Brute force optimization: In order to understand the behavior of the prototype Network and to obtain optimal results for further comparison, the model is first tested using the whole set of differential equations. On the first method, the algorithm integrates the system of differential equations, testing the possible values of \( t_{reg} \).

\[
t_{reg} = [0, \ldots, t_{final}], t_{reg} \in \mathbb{N}
\]
and determines what is the value that maximizes the final concentration of \( u_5 \). In order to show that the optimal transition is \( f(t) = 0 \rightarrow f(t) = 1 \) a simulation was run with the inverse profile (13).

\[
f(t) = \begin{cases} 
1 & \text{if } t \leq t_{reg}, \quad t \in [0, t_{final}] \\
0 & \text{if } t > t_{reg} 
\end{cases}
\]  

(13)

The second method divides \( f(t) \) in several time intervals and, without imposing any constraints on the allowed values, optimizes the function.

2) Bi-Level Optimization algorithm structure: The Bi-Level optimization algorithm was structured to accommodate missing information on the kinetics of the network. The boxed metabolites and fluxes from Fig. 1 are a part of the network that might not be fully described in terms of kinetics. The missing kinetic information is replaced with stoichiometric data and flux balance analysis is used to obtain the proper flux distribution. Thus, an inner optimization determines the fluxes during the batch time. The first step of the inner optimization process is to define the initial conditions of the input \( u_1 \) and outputs \( u_3, u_5 \). Then, a valid distribution for the fluxes \( v_1, v_2, v_3 \) and \( v_4 \) is obtained. After obtaining the flux distribution, new values for the input/outputs can be calculated by integrating their expressions in the considered time interval. During this time interval the function \( f(t) \) and the values of \( v_1, v_2, v_3 \) and \( v_4 \) are kept constant. This process is repeated from \( t = 0 \) to \( t = t_{final} \). The time interval for the integration was defined to be 1 second. The inner optimization process is shown on Fig. 2 The inner optimization process allows us to obtain the product yield, \( u_5(t_{final}) \), given a certain \( f(t) \), taking into account a valid approximation of the network dynamics over the simulation time. This inner optimization is used, in the first method, to determine:

\[
J(t_{reg}) = u_5(t_{final})
\]

On the second method, the Bi-Level Optimization, the inner optimization (linear programming problem) determines the fluxes during the batch time and is subject to an outer optimization (non-linear programming problem) that determines the optimal profile for \( f(t) \). The bi-level optimization algorithm can be represented schematically as:

- Maximize \( u_5(t_{final}) \)
- Subject to \( u_3(t) \)

(14)

subject to \( S.V = 0 \)  

(15)

3) Inner-optimization using Geometric Programming: On the first implementation of the algorithm the dynamics of the boxed metabolites from Fig. 1 are used but, following the algorithm structure, steady-state is assumed. Thus, \( \hat{x}_2 \) and \( \hat{x}_4 \) from (3) become:

\[
\frac{dx_2}{dt} = \alpha_2 u_1^{a_21} - \beta_2 v_3^{h_23} x_2^{h_22} = 0 \\
\frac{dx_4}{dt} = \alpha_4 u_3^{a_43} x_2^{a_42} (u) - \beta_4 x_4^{h_44} = 0
\]

In this implementation of the algorithm, the inner optimization problem determines the profile of the metabolites, instead of fluxes, due to the nature of the equations. The metabolite concentrations are calculated in the beginning of each time interval, solving a Geometric Programing problem, and used with (3) to integrate the values of \( u_1, u_3 \) and \( u_5 \) during that interval.

4) Inner-optimization using Linear Programming: On the second implementation it is assumed that only stoichiometric information is available for the reactions inside the box of Fig. 1. The equations of \( \dot{x}_2 \) and \( \dot{x}_4 \) become:

\[
\frac{dx_2}{dt} = v_1 - v_2 - v_3 = 0 \\
\frac{dx_4}{dt} = v_3 (u) (u) - v_4 = 0
\]

The used stoichiometric parameters are an approximation of the original network, which can be obtained from the kinetic parameters. The equation for \( \frac{dx_4}{dt} \) contains a term, \( u_3 \), that multiplies with one of the fluxes, \( v_3 \). In the context of FBA, such a term is not common but in this case it is necessary to model the forward feedback. Using FBA, the previous equations are used to calculate the distribution of fluxes. Since that the expressions for \( x_1, x_3 \) and \( x_5 \) from (3) are given in order to the metabolite concentration, the expressions can not be integrated directly. A term that relates the flux and the metabolite concentration must be
determined for each specific network. For instance, in *E. Coli* a valid relation between the product concentration variation (metabolite) and the the growth rate (flux) is $\frac{\text{Product}}{\text{dt}} = (\text{GrowthRate}) \times \text{Biomass}$ [2], the same relation applies to the Biomass variation. The current example is simpler, from 1 and 3 the fluxes $v_2$ and $v_4$, obtained with FBA, are directly used in the expressions of $u_3$ and $u_5$.

IV. RESULTS

A. Brute-Force optimization, first method, and Inner-Optimization

The first method for Brute-Force optimization took around 3min to run. Fig. 3 plots the resulting function $J(t_{reg}) = u_5(t_{final})$. It is clear from the figure that there is an optimal time of regulation to maximize the yield of $u_5$. The optimal time of regulation is $t_{reg} = 9s$. If $f(t)$ switches from 0 to 1 before $t_{reg}$ is reached the formed biomass won’t be enough to maximize $u_5(t_{final})$, on the other hand, if $f(t)$ switches from 0 to 1 after $t_{reg}$, there will be enough biomass but the time won’t be enough to produce the maximum possible amount of $u_5$. A second simulation was performed with the profile for $f(t)$ shown in (13), the obtained $u_5$ yield was always low and no optimal $t_{reg}$ was observed. The prototype network was then tested with the obtained optimal $t_{reg}$ and compared with lower and upper values in order to show that the product yield is maximum for the optimal $t_{reg}$. Fig. 4 plots $J(t_{reg}) = u_5(t_{final})$ for $t_{reg} = 4s$, $t_{reg} = 9s$ and $t_{reg} = 14s$. As expected, the function $f(t)$ with $t_{reg} = 9s$ has the higher product yield. The optimization using only the inner-optimizations took 54s and 28s using Geometric Programming and Linear Programming, respectively. Fig. 5 plots $J(t_{reg}) = u_5(t_{final})$ for the two optimizations. Comparing Fig. 5 with Fig. 3 it can be seen that the profiles remain similar. The final product yield, $u_5(t_{final})$, increases with $t_{reg}$ until the optimal value is reached, then it starts decreasing. The optimal time of regulation obtained with both GP and LP on the inner optimization was $t_{reg} = 9s$. The profile of $J(t_{reg}) = u_5(t_{final})$ with LP on the inner-optimization is not as smooth as using GP or the whole set of equations.

B. Brute-Force optimization, second method, and Bi-Level Optimization

The second method for Brute-Force optimization and the Bi-Level optimization using both GP and LP on the inner-optimization were then used to obtain $f(t)$. The simulations were run for different number of intervals. For the three cases (Brute-Force, Bi-Level with GP and Bi-Level with LP) the results were as expected and very similar. All the obtained $f(t)$ functions converged to $f(t) = 0$ when $t << t_{reg}$ and $f(t) = 1$ when $t >> t_{reg}$. The critical time point was at $t = t_{reg}$. Three cases were observed:

- The transition of $f(t)$ was $f(t) = 0 \rightarrow 1$ specially when the number of intervals was low ($< 15$) and $f(t) = u_i$ switches to $f(t) = u_{i+1}$ near $t = t_{reg}$, such as in $f(t)$ with 3, 6 or 12 intervals.
- In some cases, $f(t)$ assumes a value different than 0 or 1 during one or more time samples near $t = t_{reg}$, this happens mostly for higher number of intervals. These cases are due to convergence problems on the optimization algorithm, thus, forcing those samples to 1 or 0 will result in a higher value for $u_5(t_{final})$. Such an
example can be seen optimizing $f(t)$ with 15 intervals. The output of the Brute-Force optimization is:

$$f(t) = \{0_{i=1,2,3}, 0.07775, 0.2496, 0.8198, 1_{i=7,...,15}\}$$

Bi-Level optimization with GP outputs:

$$f(t) = \{0_{i=1,2,4}, 1, 0.7171, 1_{i=7,...,15}\}$$

and Bi-Level Optimization with LP:

$$f(t) = \{0_{i=1,...,6}, 0.9889, 1_{i=8,...,15}\}$$

Forcing the function $f(t)$ to

$$f(t) = \{0_{i=1,...,4}, 1_{i=5,...,15}\}$$

results on a higher yield for all three methods.

- Finally, in some cases where $f(t)$ assumes values different than 0 or 1, forcing those values to 0 or 1 will not increase the final yield. Although the product yield is smaller it is important to note that the difference is always a small percentage. This means that both solutions are in the optimal region of $f(t)$ and algebraic problems on the algorithm might be responsible for this problem.

V. CONCLUSIONS

For a class of networks in which the yield of the product that favors cell population growth (the “natural” product) competes with the desired product yield, with the manipulated variable affecting linearly the fluxes, it has been shown that the optimal control assumes only extreme values. The use of a bi-level optimization strategy, that maximizes the natural product in the inner level by manipulating the fluxes, leads to a good approximation to the optimal solution, with the advantage of not requiring the full knowledge of the network model. The used example network is very simple, real life networks are extremely complex and exhibit relations between metabolites that are not always expected or fully understood. This gives emphasis to the need of good in silico models and also to the determination of the exact branches to be modified when optimizing a network. The prototype network has proved useful to test the optimization strategies but a more complex network should be used to confirm that the strategy can be scaled to a bigger network.

REFERENCES

[1] Albert Sorribas, Benito Hernandez-Bermejo, Ester Vilaprinyo, Rui Alves (2006). “Cooperativity and Saturation in Biochemical Networks: A Saturable Formalism Using Taylor Series Approximations.”

[2] Gadkar, K. G., Doyle III, F. J., Edwards, J. S., & Mahadevan, R. (2005). “Estimating Optimal profiles of genetic alterations using constraint-based models”. Biotechnology and Bioengineering, 89, 243-251.

[3] K. Gadkar, R. Mahadevan and F. Doyle III (2006). Optimal genetic manipulations in batch bioreactor control. Automatica, 42:1723-1733.

[4] Anthony P. Burgard Priti Pharkya, Costas D. Maranas (2003). “Optiknock: A Bilevel Programming Framework for Identifying Gene Knockout Strategies for Microbial Strain Optimization”

[5] A. Varma and B. O. Palsson. 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, 1994.

[6] Radhakrishnan Mahadevan, Jeremy S. Edwards, Francis J. Doyle, III. Dynamic Flux Balance Analysis of Diaxixic Growth in Escherichia coli. Biophysical Journal, volume 83, September 2002, 1331-1340.

[7] Edwards JS, Covert M, Palsson B. 2002. Metabolic modelling of microbes: the flux-balance approach. Environ. Microbiol 4:133-140.

[8] Christophe H. Schilling, Jeremy S. Edwards, David Letscher, Bernhard Palsson. 2000. Combining Pathway Analysis with Flux Balance Analysis for the Comprehensive Study of Metabolic Systems.

[9] Koh K, Kim S, Mutapic A, Boyd S: GGPLAB: A simple Matlab toolbox for Geometric Programming, 2006.

[10] J. Nielsen(2001). Metabolic Engineering. Appl. Microbiol. Biotechnol. (2001) 55: 263:283. DOI 10.1007/s002530050511.

[11] Y. Liu, H. B. Sun, H. Yokota (2003). Regulating gene expression using optimal control theory. Proc. 3rd IEEE Symp. on Bioinformatics and Bioengineering (BIBE’03), Bethesda, Maryland, USA.

[12] A. Datta and E. Dougherty (2007). Introduction to genomic signal processing with control. CRC Press (Taylor & Francis Group).

[13] P. Pharkya, C. Marnas (2006). An optimization framework for inferring reaction activation/inhibition or elimination candidates for overproduction in microbial systems. Metabolic Engineering, 8:1-13.

[14] Alberto M. Sanguino, Eberhard O. Voit, C. Gonzalez-Alcon, Nestor V. Torres (2007). Optimization of biotechnological systems through geometric programming.

[15] Boyd S, Vandenberghe L (2004). Convex Optimization. Cambridge University Press.

[16] Stephen P. Boyd, Seung Jean Kim, Lieven Vandenberghe, Arash Hassibi. A Tutorial on Geometric Programming. Optimization and Engineering, 2005.