

RESEARCH ARTICLE

Open Access

Enumerating metabolic pathways for the production of heterologous target chemicals in chassis organisms

Pablo Carbonell, Davide Fichera, Shashi B Pandit and Jean-Loup Faulon*

Abstract

Background: We consider the possibility of engineering metabolic pathways in a chassis organism in order to synthesize novel target compounds that are heterologous to the chassis. For this purpose, we model metabolic networks through hypergraphs where reactions are represented by hyperarcs. Each hyperarc represents an enzyme-catalyzed reaction that transforms set of substrates compounds into product compounds. We follow a retrosynthetic approach in order to search in the metabolic space (hypergraphs) for pathways (hyperpaths) linking the target compounds to a source set of compounds.

Results: To select the best pathways to engineer, we have developed an objective function that computes the cost of inserting a heterologous pathway in a given chassis organism. In order to find minimum-cost pathways, we propose in this paper two methods based on steady state analysis and network topology that are to the best of our knowledge, the first to enumerate all possible heterologous pathways linking a target compounds to a source set of compounds. In the context of metabolic engineering, the source set is composed of all naturally produced chassis compounds (endogenous chassis metabolites) and the target set can be any compound of the chemical space. We also provide an algorithm for identifying precursors which can be supplied to the growth media in order to increase the number of ways to synthesize specific target compounds.

Conclusions: We find the topological approach to be faster by several orders of magnitude than the steady state approach. Yet both methods are generally scalable in time with the number of pathways in the metabolic network. Therefore this work provides a powerful tool for pathway enumeration with direct application to biosynthetic pathway design.

Background

Metabolism is the process of synthesis and degradation of molecules occurring in living organisms. Metabolism is generally represented as a network where metabolites are interconnected by reactions. In order to give a functional description of metabolism, metabolic networks are often decomposed into separated parts, called metabolic pathways. The description of metabolism through metabolic pathways is useful, even though any division in pathways is arbitrary, because it helps in modeling and understanding the behavior of the full network. A metabolic pathway can be defined as a coherent set of enzyme-catalyzed biochemical reactions by which a

living organism transforms a set of source compounds into a set of target compounds. By regulating enzyme and protein synthesis, living organisms can adapt to different environments. This model of metabolism as composed by independent metabolic pathways is simplistic, since pathways are nested and interdependent. In fact, metabolism is a complex system and pathways interact with each other.

The aim of the work presented here is to find all the viable sets of heterologous enzymes, which can produce a predefined target compound when added to the pool of endogenous enzymes of a given organism. Our method enables a metabolic engineer to find all heterologous metabolic pathways producing a target compound, for instance liquiritigenin (a plant secondary metabolite with therapeutic applications), from the

* Correspondence: jfaulon@gmail.com

iSSB, Institute of Systems and Synthetic Biology, University of Evry, Genopole Campus 1, Genavenir 6, 5 rue Henri Desbrières, 91030 EVRY Cedex, France

endogenous metabolites of *E. coli* K-12, as shown in Figure 1.

As depicted in Figure 1, our problem can be formulated as searching for all possible heterologous pathways linking a target compound to the endogenous metabolites of an organism. To this purpose we provide software tools that enable the discovery of potential pathways producing a target chosen by the user [1]. More precisely the user enters a target compound, a chassis organism, and our software tools return a ranked list of pathways (each list being composed of enzymes) to be engineered into the chassis organism. To achieve this task we have developed an approach composed of three steps. In the first step, using a retrosynthesis software, reactions producing a target compound are iteratively searched backwards until the set of needed precursors only contains source metabolites. This first

step returns a retrosynthetic network connecting a target compound to the endogenous metabolites of an organism. There may be several pathways in the retrosynthetic network linking the source metabolites to the target compounds and there is thus a need to enumerate all the possibilities. Pathway enumeration is performed by in the second step. Once the pathways have been enumerated, we evaluate in the third step the possibility to insert each pathway and its associated heterologous enzymes in the host organism. This step consist of determining the catalytic efficacy of the enzymes, the toxicity of the products and the coproducts [2], and the easiness of inserting the enzymes into the host. The efficiency of the pathways can then be further estimated by flux models for the cell metabolism such as flux balance analysis [3].

We have already discuss elsewhere the first and third step [1,2], i.e. methods to generate retrosynthetic networks and methods to rank pathway efficiency. To apply these methods in the context of heterologous target production, we need a computationally fast method to enumerate all possible pathways. We address the enumeration problem in the current paper.

Different mathematical models that describe metabolism have been proposed (cf. [4] for a review of the different models). We distinguish two main families of approaches: the ones computing steady states of the fluxes of reactions (one well-known application being the flux balance analysis) and the ones based only the topology of the network. Typically, steady states are studied and simulated by generating the flux space. Of particular interest are the extreme pathways and the elementary modes, they both represent the smallest (minimal) generating set of the flux space and they both are composed of independent non-decomposable pathways in the network [4]. The differences between extreme pathways and elementary modes have already been discussed in details [5] and these differences arise when dealing with reversible reactions. In the present paper we consider all reaction irreversible, and networks comprising reversible reactions are modeled by doubling each reversible reaction into a forward reaction and a reverse reaction. Algorithms have been developed to enumerate both extreme pathways [6] and elementary modes [7] and these algorithm are all variants of the double description method [8], which enumerates all extreme rays of a polyhedral cone. The algorithms use as input a stoichiometric matrix (S) representing the network (cf. [3] for definition of stoichiometric matrix) and output sets of fluxes (v) satisfying $Sv = 0$. One notices that extreme pathways and elementary modes while representing pathways (to each flux verifying $Sv = 0$ correspond a stoichiometrically balanced pathway) do not directly enumerate all pathways linking a source set

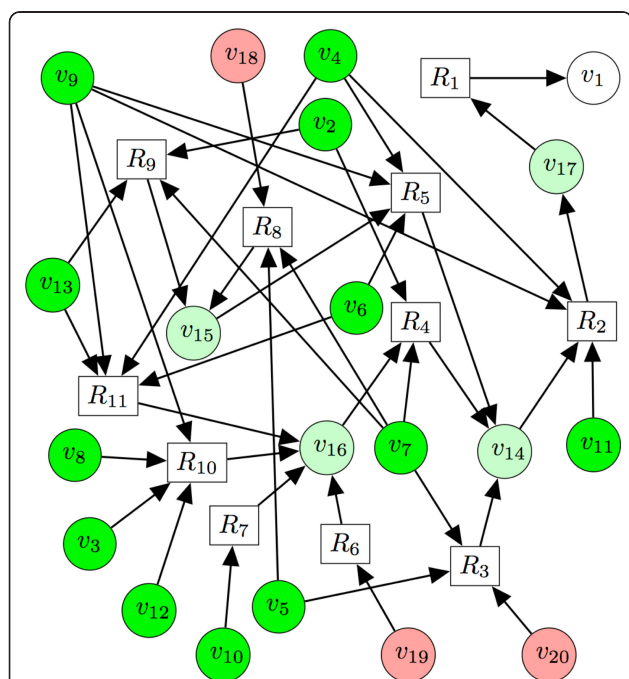


Figure 1 Known heterologous reactions leading to the production of liquiritigenin (white circular node). Squares represent reactions and circular nodes represent molecules. Dark green nodes are present in the host organism, *E. coli* K-12, light green can be produced via enzyme catalyzed reactions and red ones cannot. Liquiritigenin, a highly selective estrogen receptor β agonist, is represented by the white node v_1 . Side products not consumed in this pathway are not represented for simplicity. Four pathways lead from the host compounds to the production of the target molecule, one for instance is the pathway involving reactions R_1, R_2, R_4, R_7 . Legend: v_1 : liquiritigenin, v_2 : ATP, v_3 : NADH, v_4 : NADPH, v_5 : NADP+, v_6 : oxygen, v_7 : CoA, v_8 : acetate, v_9 : H+, v_{10} : L-tyrosine, v_{11} : malonyl-CoA, v_{12} : 4-hydroxybenzoate, v_{13} : trans-cinnamate, v_{14} : p-coumaroyl-CoA, v_{15} : cinnamoyl-CoA, v_{16} : 4-coumarate, v_{17} isoliquiritigenin, v_{18} : cinnamaldehyde, v_{19} : 3-(4-hydroxyphenyl)lactate, v_{20} : 4-hydroxycinnamyl aldehyde.

to a target set of compounds. However as shown in the subsection “Enumerating pathways using the steady state approach” one can construct stoichiometric matrices where input fluxes are added to the set of source compounds and outgoing fluxes are associated to the target and heterologous coproducts such that the extreme pathways and elementary modes enumerated from these matrices do correspond to all pathways linking the source set to the target.

While as mentioned above, the problem of systematically enumerating pathways for heterologous production in chassis organisms has not yet been addressed, there are methods based on the steady state approach to search for heterologous pathways optimizing target productions [9], and methods to search for shortest pathways between source and target sets of compounds [10] and [11]. All these methods are based on optimization and make use of integer linear programming. Precisely, the method of Pharkya *et al.* [9], is aimed at redesigning microbial chassis organisms through heterologous reaction addition and native reaction deletion for the overproduction of a target compound. The addition and deletion are parameterized using binary variables attached to each reaction. A mixed integer linear program (MILP) is then set up to maximize the target yield while minimizing the number of added reactions. The methods of de Figueiredo *et al.* [10], and Pey *et al.* [11] are both aimed at searching for the k shortest pathways. In de Figueiredo *et al.* [10] the k first shortest pathways are searched in entire metabolic networks, while in Pey *et al.* [11] the pathways are searched between a source metabolite and a target metabolite. Both methods solve the problem at steady state and search for fluxes, v , verifying $Sv = 0$, while minimizing the number of reactions turned on (using a binary variable). Aside from the fact that integer linear programs suffer from computational complexity (MILP is an NP-hard problem) all the above methods search for at most k optimized (shortest) pathways and do not guarantee a full enumeration of the possibilities. In our methods the optimal pathways are computed in a post process by ranking the pathways that have been enumerated. Our approach allows one to decouple enumeration from optimization, and thus to plug any optimization criteria, including nonlinear functions and not only target yield or pathway length (cf. page 3 and Carbonell *et al.* [1] for a list of criteria entering our metabolic engineering optimization problem).

Aside from using extreme pathways and elementary modes, we also present in this paper a topological model which directly enumerates all the possible heterologous pathways linking target compounds to a source set of compounds. The main advantage of the topological approach compared to the stationary state approach is computational speed. Speed is in fact an important

aspect when searching for the best pathways to engineer, as there are generally a combinatorial number of pathways between given source sets and target sets. As an illustration of this combinatorial complexity, the work of Hatzimanikatis *et al.* [12], which provides a list of 75,000 novel biochemical routes from chorismate to phenylalanine, and the work of Cho *et al.* [13], which enumerates 107,272 reaction routes to produce isobutanol.

There exist standard graph-based methods to search and eventually enumerate pathways in metabolic networks, but these methods including PathFinding [14-16] and Pathway Hunter Tool [17] are computing pathways and shortest pathways in graphs instead of hypergraphs. The particularity of these techniques is that only main substrates and main products are taken into account when constructing pathways, and consequently these main compounds must be differentiated from the cofactors (i.e. co-substrates and co-products). In the work of Croes *et al.* [14,15] cofactors are filtered out based on their connectivity in the network. Indeed, compounds highly connected such as ATP, NADP, or H_2O are cofactors of most reactions as they do not share carbon atoms with the products of the reactions. In a more recent work [16], the main compounds in the pathways linking source metabolites to target metabolites are detected using the Kegg RPAIR annotation [18,19], which enables one to follow the fate of atoms when going from a set of substrates to a set of products. Another approach to search for main substrates and main product is the one developed with the Pathway Hunter Tool, which consists of mapping substrates to products using cheminformatics fingerprints. While all the above techniques are computationally efficient, their main shortcoming is that they are not able to encompass reactions when a main product is formed from two main substrates. There are plenty of such reactions in metabolic networks, consider for instance the formation of guanidinoacetate from arginine and glycine through a glycine amidinotransferase (EC 2.1.4.1), or the formation of glutathione from γ -L-glutamyl-L-cysteine and glycine catalyzed by a glutathione synthase (EC 6.3.2.3). Recently, some of the above topological methods have been benchmarked against the integer linear programming technique mentioned above [11] to search for the shortest pathways linking various compounds, the recovery ratio for a set of 40 predefined reference pathways could not reach 100% with the graph based approach, exemplifying the shortcoming of that approach.

As reviewed above, while there are methods and theoretical results to enumerate elementary modes or extreme pathways and graph based techniques to search for pathways in a given metabolic network, to the best of our

knowledge there is no known methods to directly enumerate pathways in the context of metabolic engineering, that is, to enumerate all the pathways encompassing all substrates and products necessary and sufficient to produce a given set of target compounds from a given set of source compounds. In the present paper we address that specific problem and present two methods one based on elementary modes (steady state approach) and one based on a direct enumeration algorithm (topological approach). In order to address this problem we need, in addition, to consider the problem of determining supplement molecules, i.e. metabolites that the organism cannot synthesize, but which can be added to the growth media in order to increase the number of viable pathways; and bootstrap molecules, i.e. metabolites which are required first in order to be produced [20]. While in the general context of metabolic network analysis, finding elementary modes does not require to first search for bootstrap molecules, in the context of metabolic engineering however any heterologous pathway solution that comprises a compound that is first consumed before being produced is valid only when the compound is added to the growth medium. Therefore, in our study in the context of metabolic engineering, there is a need to first compute the bootstrap compounds prior to elementary modes.

The paper is divided as follows. In the Methods section we first provide some definitions, then outline our algorithms to solve the pathway enumeration problem with both the steady state approach and the topological approach. The problem of finding and enumerating all the pathways going from a large source (as for instance all the metabolites of an organism) to a target chosen by the user is considered. All the algorithms presented for the topological approach (with the exception of the algorithm for enumeration) have polynomial worst-case running time, the algorithm for enumeration is a polynomial time per output algorithm on some classes of hypergraphs. We also provide algorithms to determine supplements, which are metabolites that the organism cannot synthesize, but which can be added to the growth media in order to increase the number of viable pathways. Furthermore, an analysis of pathways containing supplements allows finding out pathways that contain bootstrap molecules. In the Results and Discussion section we illustrate our algorithms with the enumeration of the possible pathways to synthesize more than 5000 compounds in *E. coli*. We “experimentally” probe the computational complexity of the steady state and topological approaches for a series of networks of growing sizes and discuss the theoretical complexity results of the topological approach, which are provided in Appendices A and B.

While we illustrate our two methods for the production of heterologous compounds using as a source set all the endogenous metabolites of *E. coli*, our methods can be applied to any chassis organism and more generally to any source set (for instance a set of nutrients or a set of abundant currency metabolites).

Methods

In the context of metabolic engineering, metabolic networks have been represented as directed graphs (cf. for instance Cho *et al* [13]). In such graphs, edges are directed and correspond to reactions connecting two compounds if one is the product of the other. Directed graphs can represent monomolecular reactions (one substrate gives one product), but they are not well suited to capture more complex reactions. As already discussed in the background section, when representing bimolecular reactions, one has to choose which molecules are connected by the edges of the graph and which ones have to be excluded from the graph because they are considered co-substrates or co-products. Additionally, one of the limitations of a model based on a graph representation is that depending on the criteria used to identify the co-substrates and co-products in the reactions, the networks obtained are different.

In the present paper to palliate the limitations of the directed graph model, we represent networks as directed hypergraphs. The first examples of modeling through hypergraphs can be found in [20]. In a hypergraph, each hyperarc connects a set of vertices, corresponding to reactants, to a disjoint set of vertices, representing the products. In our model each hyperarc corresponds to a reaction that can be catalyzed by an enzyme. It is worth noticing that hypergraph models have already been used to find minimal sets of metabolites sufficient to produce a set of target metabolites [21]. Unfortunately, the algorithms proposed in [21], do not enumerate pathways and are therefore not directly applicable to our metabolic engineering problem.

Definitions

Definition 1 (Hypergraphs and hyperarcs).

A directed hypergraph is a pair $\mathcal{H} = (V, E)$ where $V = \{v_1, v_2, \dots, v_n\}$ is the set of vertices and $E = \{e_1, e_2, \dots, e_m\}$ is the set of hyperarcs. A hyperarc e_i is an ordered pair $e_i = (X_i, Y_i)$ of disjoint subsets of V .

The set X_i is also called the tail of e_i and the set Y_i is called the head, with reference to the graphical representation of arcs (directed edges) and hyperarcs as arrows.

We denote by $X : E \rightarrow \mathcal{P}(V)$ the application that given an hyperarc e_i returns its tail $X(e_i) \subset V$.

Analogously we use $Y : E \rightarrow \mathcal{P}(V)$ for the application that given a hyperarc returns its head.

Definition 2 (Reactions and networks).

In a metabolic network each vertex corresponds to a metabolite and each hyperarc corresponds to a reaction. A metabolic network of m metabolites and n reactions can be represented with a $m \times n$ stoichiometric matrix S , where the rows correspond to the m metabolites and the n columns to the reactions. A reaction j is represented by the column vector $S_j = (s_{1j}, \dots, s_{mj})^T$ where s_{ij} is the stoichiometric coefficient of metabolite i in reaction j . Reactants have negative coefficients and products have positive coefficients.

Examples of hypergraph, network, and stoichiometric matrix are given in Figure 2A. We notice that the stoichiometric coefficients of the reactions are not taken into account in the hypergraph representation. We also notice that the pair (X, Y) is ordered so to make the distinction between reactants and products. In this representation reactions are irreversible. Many biochemical reactions can be considered as irreversible, since in organisms the homeostatic equilibrium is often strongly polarized. Nonetheless, metabolic network may comprise reversible reactions, and we model these reactions by introducing both hyperarcs: (X, Y) , and (Y, X) .

Hyperpaths, a generalization of simple paths in graphs where cycle free paths going from one vertex to another, are used to represent pathways. A hyperpath connects a source set of vertices to a target set of nodes. Two examples of hyperpaths are given in Figures 2A and 2C. We remark that in a natural way a set E of hyperarcs defines a hypergraph $\mathcal{H} = (\cup_{e \in E} X(e) \cup \cup_{e \in E} Y(e), E)$. By abuse of the terminology we denote by E the hypergraph corresponding to the set E of hyperarcs and all the heads and tails of the hyperarcs in E . The following definition for hyperpaths is borrowed from Nielsen et al. [22].

Definition 3 (Hyperpaths).

A hyperpath P going from a source subset $S_{\mathcal{H}}$ of V to a target subset T_P of P in a hypergraph $\mathcal{H} = (V, E)$ is a hypergraph $\mathcal{H}_P = (V_P, E_P)$ with $V_P \subseteq V$, $E_P \subseteq E$, such that there is an ordering F of the hyperarcs E_P with the following properties.

- $\forall k \in \{0, \dots, |F|\}, X(F_k) \subseteq S_{\mathcal{H}} \cup (\cup_{j < k} Y(F_j))$
- $T_P \subseteq S_{\mathcal{H}} \cup (\cup_{e \in E_P} Y(e))$

From the point of view of metabolism, the first condition corresponds to the requirement that reactants of reactions participating in the hyperpath can be produced without the presence of the reaction itself. Hyperpaths defined in this manner represent a metabolic route from

the source to the target. According to definition (3) the hypergraph of Figure 2B with source a is not a hyperpath because neither reaction R_1 nor R_2 can happen until the other does not start. The definition (3), though complex, is computationally tractable, meaning that the time required to determine if a hypergraph is a hyperpath is proportional to the number of reactions. A polynomial time algorithm to determine if a hypergraph is a hyperpath is given in [23], the algorithm *FindAll* presented below can also be used for that purpose. In fact, as discussed below, if the set of reactions returned by *FindAll* $(\mathcal{H}_P, S_{\mathcal{H}})$ contains all the reactions in \mathcal{H}_P , then \mathcal{H}_P is a hyperpath.

The metabolic network described by a hypergraph has to be as comprehensive as possible, containing every known enzyme-catalyzed reaction occurring in organisms. We say that a hyperpath produces a set of target metabolites if it contains all those target elements. A set of target compounds is said to be reachable from a given source, or linked to the source, if there is at least one hyperpath producing the targets.

We are interested in the enumeration of pathways leading to the production of a desired compound. Hyperpaths do not generally give the best representation of pathways because hyperpaths can contain reactions not necessarily linking the target to the source. Minimal hyperpaths, cf. definition (4), are an appropriate representation of pathways since they contain only the essential reactions linking the source to the target.

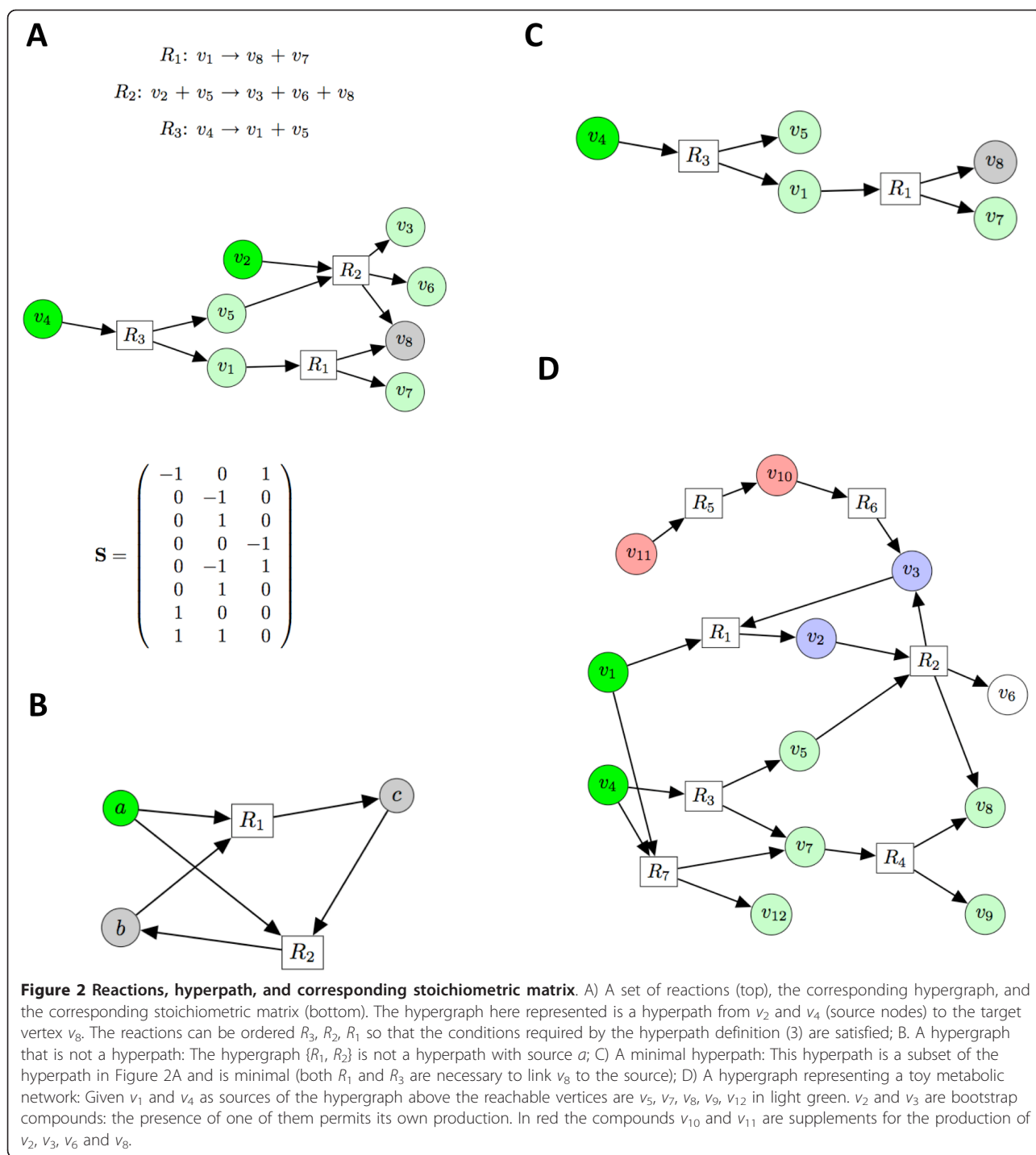
In the definition given below, we say that a hyperpath $\mathcal{P}(V, E)$ is a subset of another hyperpath $\mathcal{P}'(V', E')$ if $V \subseteq V'$ and $E \subseteq E'$. For instance the hyperpath of Figure 2C is a subset of the one of Figure 2A.

Definition 4 (Minimal Hyperpaths).

A hyperpath (V_P, E_P) with target T_P is said to be minimal if it has no proper subsets with the same target.

The target is disconnected from the source if a reaction is removed from a minimal hyperpath. In this sense minimal hyperpaths cannot be reduced. From a metabolic engineering perspective the concept of minimal hyperpath is useful as it defines the minimum set of reactions necessary to produce a target heterologous compounds, and consequently the minimum set of enzymes needed to be inserted into the chassis organism where the compound is going to be produced.

In the following we define $B(\mathcal{H}, S_{\mathcal{H}})$ to be the set of all molecules linked to the source for a given hypergraph \mathcal{H} and source set $S_{\mathcal{H}}$. The characterization of $B(\mathcal{H}, S_{\mathcal{H}})$ is the first task to be solved before the enumeration. Once this set is known all the minimal hyperpaths can be enumerated for all the molecules associated to the vertices in $B(\mathcal{H}, S_{\mathcal{H}})$.



Supplements

Supplements for a target are molecules whose presence in the source set increases the number of pathways for target production. Finding supplements is an important improvement when exploring ways to produce the target, since they make possible new pathways.

For each target of interest one can look for vertices that once inserted in $S_{\mathcal{H}}$ give place to pathways otherwise impassable. In terms of metabolism we are looking for the “supplement” molecules, i.e., molecules that once introduced in the source set permit to find more pathways than those otherwise available. We introduce

below *FindSupp*, an algorithm that returns the supplements.

An analysis of pathways containing supplements allows to find out pathways containing bootstrap molecules, i.e. metabolites that are needed in reactions producing compounds afterwards used for the production of the bootstrap molecules. As a matter of fact, many pathways can be made viable once bootstrap molecules become available in the metabolic network (a concept introduced in [20]). Loosely speaking bootstrap molecules are molecules that cannot be produced by the reactions belonging to a hyperpath unless they are already present in the source. Cottret *et al* [21] stated that given a source set the existence of a pathway making use of bootstrap molecules can be tested in polynomial time. We provide later in this section an algorithm returning the bootstrap compounds, such algorithm can be used to determine if a target molecule is connected to the source through a pathway making use of bootstraps.

Enumerating pathways using the steady state approach

In steady state, all possible pathways in a metabolic network are by definition stoichiometrically balanced, i.e. all metabolites produced from the source set must be consumed except for those that are target products. Extreme pathways and elementary modes are two methods that compute the set of independent non-decomposable pathways in the network that generate all feasible steady state solutions in the flux space. They do not directly enumerate all pathways linking a source set to a target set of compounds. However, one can construct stoichiometric matrices where input fluxes are added to the set of source compounds and outgoing fluxes are associated to the target and heterologous co-products such that the extreme pathways and elementary modes enumerated from these matrices can be used to generate all pathways linking the source set to the target.

Given a hyperpath $\mathcal{H}_p = (V, E)$ of a hypergraph $\mathcal{H} = (V, E)$, we can define a set of flux vectors \mathbf{v}_p for the hyperpath where components v_{pj} corresponding to those reactions in the pathway $e_j \in \mathcal{H}_p$ are activated:

$$v_{pj} = \begin{cases} > 0 & e_j \in \mathcal{H}_p \\ 0 & e_j \in \mathcal{H} \setminus \mathcal{H}_p \end{cases} \quad (1)$$

A hyperpath $\mathcal{H}_p = (V, E)$ of a hypergraph $\mathcal{H} = (V, E)$ with input source subset $S_{\mathcal{H}}$ and the target subset T_p is defined as stoichiometrically balanced if the rows corresponding to each metabolite $v_i \in V$ that are obtained from the product of the stoichiometric matrix \mathbf{S} and the associated flux vector \mathbf{v}_p verify:

$$\mathbf{S}\mathbf{v}_p = \begin{cases} \leq 0 & v_i \in S_{\mathcal{H}} \\ \geq 0 & v_i \in T_p \\ \mathbf{0} & v_i \in V \setminus \{S_{\mathcal{H}}, T_p\} \end{cases} \quad (2)$$

A way to introduce the constraint on input and output metabolites in the previous equation is by adding to the stoichiometric matrix \mathbf{S} additional columns corresponding to input reactions (reactions with no substrate that produce the source set $S_{\mathcal{H}}$), and output reactions (reactions with no product that consume the product metabolites in the hypergraph T_p). These auxiliary reactions, even if non-properly balanced in terms of the law of conservation of mass, are useful in order to define completely the problem in a compact manner:

$$\begin{aligned} \mathbf{S}\mathbf{v} &= \mathbf{0} \\ \mathbf{v} &\geq \mathbf{0} \quad \mathbf{v} \in \mathcal{R} \end{aligned} \quad (3)$$

Both extreme pathways and elementary modes make use of this formulation in order to compute the set of feasible solutions \mathbf{v} . Since in our hypergraph definition all reactions are irreversible, the set of pathways solving Equation 3 computed by both extreme pathways and elementary modes are identical (cf. [5]). Furthermore, solutions in \mathbf{v} must contain only positive or null fluxes.

In order to determine all stoichiometrically balanced heterologous pathways \mathcal{H}_p that can be inserted into the chassis organism to produce a target set T_p , we need to constrain the computation of elementary modes only to those that have non-zero fluxes for heterologous reactions. Efficient solutions to this problem have been considered in the divide-and-conquer approach [24,25] by rearranging the constraints in an echelon form so that the constraints containing only the desired reactions appear at the bottom. To define the constraints in our case, we consider first the hypergraph \mathcal{R}_T that is formed only by heterologous reactions. This hypergraph \mathcal{R}_T is the subset of the hypergraph $\mathcal{R}(V, E)$ formed by those hyperedges where at least one vertex V does not belong to the source set $S_{\mathcal{R}}$, i.e. those metabolites endogenous to the chassis organisms. By considering \mathcal{R}_T instead of the full hypergraph \mathcal{R} , we are looking only for biosynthetic pathways involving heterologous reactions and therefore avoiding cycles internal to the chassis organism. Therefore, to compute all feasible steady state heterologous pathways, we reformulate Equation 2 so that the stoichiometric matrix \mathbf{S} is defined by reactions in \mathcal{R}_T ; the input is given by all substrates in the source set $S_{\mathcal{R}} \cap X(E_{\mathcal{R}})$; and the output by all products of the reactions in the hypergraph $Y(E_{\mathcal{R}})$.

Finally, from the computed set of solutions \mathbf{v} for Equation 3, we are interested in enumerating all minimal hyperpaths from $S_{\mathcal{R}}$ to the target set T on the

hypergraph given by \mathcal{R}_T . According to Definition 4, minimal hyperpaths for some target T are given by those cycle-free solutions in \mathbf{v} containing only reactions linking the source to the target. Since any feasible flux pattern \mathbf{v} is a superposition of elementary modes with non-negative coefficients [26], the set of minimal hyperpaths for a given target T is a subset of the elementary modes producing T that are solution of Equation 3. Namely, any feasible solution generated from the elementary modes will contain at least as many reactions as the ones that are in those elementary modes that form its basis. Therefore no additional minimal hyperpaths can be generated in this case by superposition of elementary modes.

Enumerating pathways using the topological approach

The algorithm *FindAll* that allows to find $B(\mathcal{H}, S_{\mathcal{H}})$, the set of metabolites that can be linked to the source $S_{\mathcal{H}}$ by a hyperpath. *FindAll*, by explicitly constructing the ordered set F in definition (3), provides a proof of the tractability of the problem of checking if a hypergraph is a hyperpath. Moreover *FindAllF* permits to prune the original hypergraph enabling a faster enumeration algorithm.

As presented below the algorithm *Minimize*, when called on the output of *FindAll*, returns, if exists, a minimal hyperpath linking a given target to the source. These algorithms are the main components of the algorithm enumerating the pathways *FindPath* described next. Then we present *FindSupp* an algorithm to enumerate supplements.

Finding one minimal hyperpath

Let $\mathcal{H} = (V, E)$ be the hypergraph representing the set of metabolic reactions, $n = |V|$, $m = |E|$ and let $S_{\mathcal{H}}$ be the set of source vertices representing the source metabolites.

The algorithm *FindAll* returns all the reactions that can contribute to the production of any element in $B(\mathcal{H}, S_{\mathcal{H}})$, i.e., the set of all compounds that can be connected to the source. *FindAll* is a linear algorithm in the number of vertices, hyperarcs and in the total coordination; the complexity is $O(n + m + \sum_{v \in V} |X^{-1}(v)| + |Y^{-1}(v)|)$ that is bounded by $O(n + m + n \cdot m)$. Therefore, such algorithm can be applied to the hypergraph \mathcal{H} of all reactions in order to obtain a pruned sub-hypergraph $\mathcal{H}' = (V', E')$ where the set of vertices $V' := S_{\mathcal{H}} \cup B$, and the set of edges E' is the set of reactions returned by *FindAll*. In the context of metabolic engineering *FindAll* returns all the compounds that can be produced from a given set of source compounds and reactions. For instance, using *FindAll* with all know metabolic reactions one can determine all the compounds that can be produced from the metabolites of *E. coli*.

Algorithm FindAll (Given a hypergraph \mathcal{H} and a source $S_{\mathcal{H}}$, returns all the hyperarcs that are part of at least one hyperpath.)

input: $\mathcal{H}, S_{\mathcal{H}}$

1. **for all** r **in** \mathcal{H}
2. $x(r) \leftarrow X(r)$
3. **end for**
4. $V \leftarrow S_{\mathcal{H}}$
5. $D \leftarrow S_{\mathcal{H}}$
6. $F \leftarrow \{\emptyset\}$
7. **while** $V \neq \{\emptyset\}$
8. let i be an element of V
9. $V \leftarrow V \setminus i$
10. $D \leftarrow D \cup i$
11. **for all** $r \in H$ such that $i \in x(r)$:
12. $x(r) \leftarrow x(r) \setminus i$
13. **if** $x(r) = \{\emptyset\}$
14. $F \leftarrow \{F, r\}$
15. **for all** j **in** $Y(r)$ **and not in** D :
16. $V \leftarrow V \cup j$
17. **end for**
18. **end if**
19. **end for**
20. **end while**

output:

F

Let D be the union of the source set and of the heads of all the reactions output in F by *FindAll*. The correctness of the algorithm above is given by the following claims: every element in D is the target of some hyperpath or is part of the source, and every vertex in \mathcal{H} that can be reached from the source is in D . For the first claim we can give a constructive proof by using the output vector F , the second claim is proved by contradiction.

- The proof of the fact that every element in D is reachable from the source is given constructively by the ordered set F returned by the algorithm. In fact at each step F is a hyperpath. This claim can be proved by induction on the steps of the algorithm, each time a hyperarc r is appended to F (line 14) the tail $X(r)$ is contained in D (hyperpath by inductive hypothesis) and if a vertex j is added to D (line 10) it means that it was previously added to V (line 16) and thus it was in the head $Y(E)$ of some hyperarc already in the hyperpath.
- The second claim can be proved by contradiction: if an element of $B(\mathcal{H}, S_{\mathcal{H}})$ were not in D there would be a hyperpath linking it to the source. In such hyperpath let consider the first (according to the order given by the definition) reaction r whose $X(r)$ belongs to D and such that one of the elements

of $Y(r)$ does not. Consider among $x(r)$ the last one that has been inserted into the set V ; after its removal from $X(r)$ this set becomes empty and the elements of $Y(r)$ are inserted into V (line 16) and then in D (line 10), which is a contradiction.

From the above statements follows that each vertex appearing in a hyperpath having as source $S_{\mathcal{H}}$ is an element of D and every hyperarc is an element of F . Thus the algorithm *FindAll* provides an effective pruning of the original hypergraph: in \mathcal{H} there is no minimal hypergraph with source $S_{\mathcal{H}}$ containing hyperarcs not in $\cup_k F_k$ or vertices not in $B(\mathcal{H}, S_{\mathcal{H}})$. The output hyperarcs are the only ones that can belong to a minimal hyperpath, and $\mathcal{H}' = (S_{\mathcal{H}} \cup (\cup_k Y(F_k)), \cup_k F_k)$ is the pruned hypergraph only containing reachable vertices and hyperarcs.

Notice that *FindAll* algorithm as presented above returns in polynomial time a hyperpath valid for each target vertex in $B(\mathcal{H}, S_{\mathcal{H}})$. Even though there are more efficient algorithms for finding a hyperpath for one single target, for the sake of simplicity we avoid to introduce here an additional algorithm and just remark that since *FindAll* is polynomial, the use of it does not affect the complexity analysis of the algorithms making use of its output.

Remark that a minimal hyperpath going to a specific target can be easily extracted from the hyperpath output of *FindAll*. Namely, given a hyperpath \mathcal{P} connecting S to T , it is always possible to find a minimal hyperpath \mathcal{P}' subset of \mathcal{P} . Moreover it can be done in polynomial time, for instance by using *Minimize* ($\mathcal{P}, \{\emptyset\}, T, S$), the algorithm introduced below.

Minimize (\mathcal{P}, R_f, T, S) is an algorithm that takes as input a hypergraph \mathcal{P} , a hyperpath R_f subset of \mathcal{P} , a target set of vertices T and a source S . If \mathcal{P} does not link T to S the empty set is returned, otherwise a hyperpath contained in \mathcal{P} , containing R_f and linking T to S is returned. In particular, if R_f is empty, the output of *Minimize* is the minimal hyperpath going from S to T , provided it exists. *Minimize* returns a hyperpath obtained by removing all inessential hyperarcs except for the ones in R_f . In the context of metabolic engineering, pathways containing a small number of heterologous reactions are generally preferred, since they are easier to engineer in the host organism. Therefore, given two pathways that produce the same target, where one is subset of the other, the one requiring the smaller number of heterologous reactions has to be selected. This is the reason that makes relevant to obtain minimal hyperpaths from generic hyperpaths.

Algorithm Minimize (Given a hypergraph \mathcal{P} containing R_f , returns either a hyperpath from S to T containing R_f or an empty set if T is not linked to S by \mathcal{P} .)

input: \mathcal{P}, R_f, T, S
 1. $F \leftarrow \text{FindAll}(\mathcal{P}, S)$
 2. $\mathcal{P}' \leftarrow \mathcal{P}$
 3. **if** $T \not\subseteq \cup_k Y(F_k)$
 4. $\mathcal{P}' \leftarrow \{\emptyset\}$
 5. **else**
 6. **for all** r **in** \mathcal{P}
 7. **if** r **not in** R_f
 8. $F \leftarrow \text{FindAll}(\mathcal{P}' \setminus r, S)$
 9. **if** $T \subseteq \cup_k Y(F_k)$
 10. $\mathcal{P}' \leftarrow \mathcal{P}' \setminus r$
 11. **end if**
 12. **end if**
 13. **end for**
 14. **end if**
output:
 \mathcal{P}'

The proof of correctness of this algorithm is simple and is based on the fact that $\mathcal{P}' \subseteq \mathcal{P}$ implies $\text{FindAll}(\mathcal{P}', S) \subseteq \text{FindAll}(\mathcal{P}, S)$. If a reaction in \mathcal{P} has not been removed from \mathcal{P}' , then any subset of \mathcal{P}' not containing r does not produce the target. The worst-case time for this algorithm is $O(m \cdot (n + m + \sum_{r \in \mathcal{P}} |X(r)| + |Y(r)|))$. Since $X(r)$ and $Y(r)$ have bounded values, the algorithm has a quadratic complexity. Even though faster algorithms can be designed, here we presented this one because of its conceptual simplicity. Remark that, since *Minimize* ($\mathcal{P}, \{\emptyset\}, T, S$) returns a minimal hyperpath \mathcal{P}' subset of \mathcal{P} if it exists, then the minimality of a hyperpath \mathcal{P} can be tested by checking whether $\mathcal{P}' = \mathcal{P}$ or not.

A related problem to *Minimize* is the minimal constrained hyperpath problem: the problem of finding if a minimal hyperpath from a given source to a given target, containing the hyperarcs in R_f exists. Notice that *Minimize*, although linked to this problem does not solve it. In fact, if the output of *Minimize* is an empty set then there are no minimal hyperpaths satisfying the constraints; however if the output is a minimal hyperpath then obviously a minimal constrained hyperpath exists; and finally, if the output is a non-minimal hyperpath then we do not know if a minimal hyperpath satisfying the constraints exists or not.

Below we will discuss why we are interested in algorithms for the minimal constrained hyperpath problem, while in Appendix A.2 we show that in general the problem is NP-complete (reduction to 3-SAT).

Pathways Enumeration

The basic idea behind the enumeration algorithm presented below is to introduce an iterative refinement of

partitions of the space of feasible solutions i.e. of the space of hyperpaths and in each part to look for a solution. In our implementation, a part is defined by two sets of reactions (R_f and R_n) of the original hypergraph. These sets are used during the enumeration process, R_f is a set of hyperarcs that must be present in the enumerated hyperpath and R_n is the set of hyperarcs that must not be part of the enumerated hyperpath. The problem of finding a solution in one of the parts is addressed at each iteration and if a solution is found the part is divided in finer parts. This process is repeated until all the minimal hyperpaths have been found.

Enumeration by means of the minimal constrained hyperpath problem

First we describe informally the enumeration algorithm through the toy example hypergraph in Figure 2D and 3A, then we outline in Figure 3B a typical run for a more involved example: liquiritigenin (cf. Figure 1).

A minimal hyperpath \mathcal{P}_1 connecting the node v_8 to the source nodes v_1, v_4 on the hypergraph \mathcal{H} of Figure 2D can be obtained by calling *Minimize* (\mathcal{P}' , $\{\emptyset\}$, $\{v_8\}$, $\{v_1, v_4\}$) on the hypergraph \mathcal{P}' obtained by *FindAll*(\mathcal{H} , $\{v_1, v_4\}$). The hypergraph \mathcal{P}' is represented in Figure 3A.

Once $\mathcal{P}_1 = \{R_4, R_3\}$ has been obtained, the search space is divided into three parts:

- the hypergraphs which do not contain R_4 ,
- the hypergraphs which do contain R_4 and do not contain R_3 ,
- the hypergraphs which do contain R_4 and R_3 .

The first set does not contain hyperpaths connecting the target to the source: once the reaction R_4 is removed, v_8 is disconnected from the source. The second set contains a solution and thus has to be partitioned. The third set contains only one minimal pathway (the one consisting of hyperarcs R_3, R_4 highlighted in Figure 3A).

The minimal hyperpath in the second set is found by running *FindAll* on $\mathcal{H} \setminus R_3$ and then *Minimize* with constraint $R_f = \{R_4\}$. The minimal hyperpath so obtained is the one only containing hyperarcs R_4, R_7 . The set of the hypergraphs defined by ($R_f = \{R_4\}$, $R_n = \{R_3\}$) is partitioned in two parts defined by new sets of constraints. The way the partition is done is explained in detail in algorithm *FindPath* and gives two non overlapping sets:

- the hypergraphs which do contain R_4 and do not contain R_3 and R_7 ,
- the hypergraphs which do contain R_4 and R_7 and do not contain R_3 .

The first of these sets does not contain hyperpaths going to v_8 : once R_3 and R_7 are removed, node v_8 is disconnected from the source. The second one only contains the second and last minimal hyperpath: the one consisting of hyperarcs R_7, R_4 . The algorithm here sketched is based on the fact that all minimal hyperpaths are found once the problem of finding a minimal hyperpath has been solved for each part of the partition.

Relaxed hyperpath minimization

The enumeration procedure is performed by the algorithm *FindPath*, which enumerates all the minimal pathways and does not output duplicate hyperpaths. Precisely *FindPath* (\mathcal{H} , R_f , T , $S_{\mathcal{H}}$) returns a set of hyperpaths containing all the minimal hyperpaths in \mathcal{H} connecting T to $S_{\mathcal{H}}$ and containing all the reactions in R_f . *FindPath* (\mathcal{H} , $\{\emptyset\}$, T , $S_{\mathcal{H}}$) returns all the minimal hyperpaths from the source $S_{\mathcal{H}}$ to the target T in \mathcal{H} .

A schematic representation of how *FindPath* works for the enumeration of the pathways of liquiritigenin is given in Figure 3B where we represent each call with a box connected by an arrow to its parent process. For each call of *FindPath* either a new hyperpath is found and then *FindPath* is executed with new constraints, or there are no new hyperpaths and the branching process is stopped. The new constraints sets R_f' , R_n' for a new call of *FindPath* are obtained by incrementing the sets R_f , R_n of the father process. Given an order for the hyperarcs of the hyperpath \mathcal{P} found for the father process, the set R_n' relative to the child process is constructed by incrementing R_n by one element r belonging to \mathcal{P} , the set R_f' is constructed by incrementing R_f by all the hyperarcs coming before r . For each element in \mathcal{P} not belonging to R_n a child process is called.

FindPath (\mathcal{H} , $\{\emptyset\}$, T , $S_{\mathcal{H}}$) returns all the minimal hyperpaths from the source $S_{\mathcal{H}}$ to the target T in \mathcal{H} . In the context of metabolic engineering *FindPath* returns all the metabolic pathways for the production of the target compounds.

Algorithm FindPath (Enumerate all minimal hyperpaths from $S_{\mathcal{H}}$ to the target set T with constraints R_f on the hypergraph given by \mathcal{H})

input:

\mathcal{H} , R_f , T , $S_{\mathcal{H}}$

1. $F \leftarrow \text{FindAll}(\mathcal{H}, S_{\mathcal{H}})$
2. $\mathcal{P} \leftarrow \text{Minimize}(\cup_k F_k \cup R_f, R_f, T, S_{\mathcal{H}})$
3. $En \leftarrow \emptyset$
4. **if** $\mathcal{P} \neq \emptyset$
5. $En \leftarrow \mathcal{P}$
6. $F \leftarrow \text{FindAll}(\mathcal{P}, S_{\mathcal{H}})$
7. **for all** k **in** $|F|, \dots, 1$
8. $r = F_k$
9. **if** r **not in** R_f :

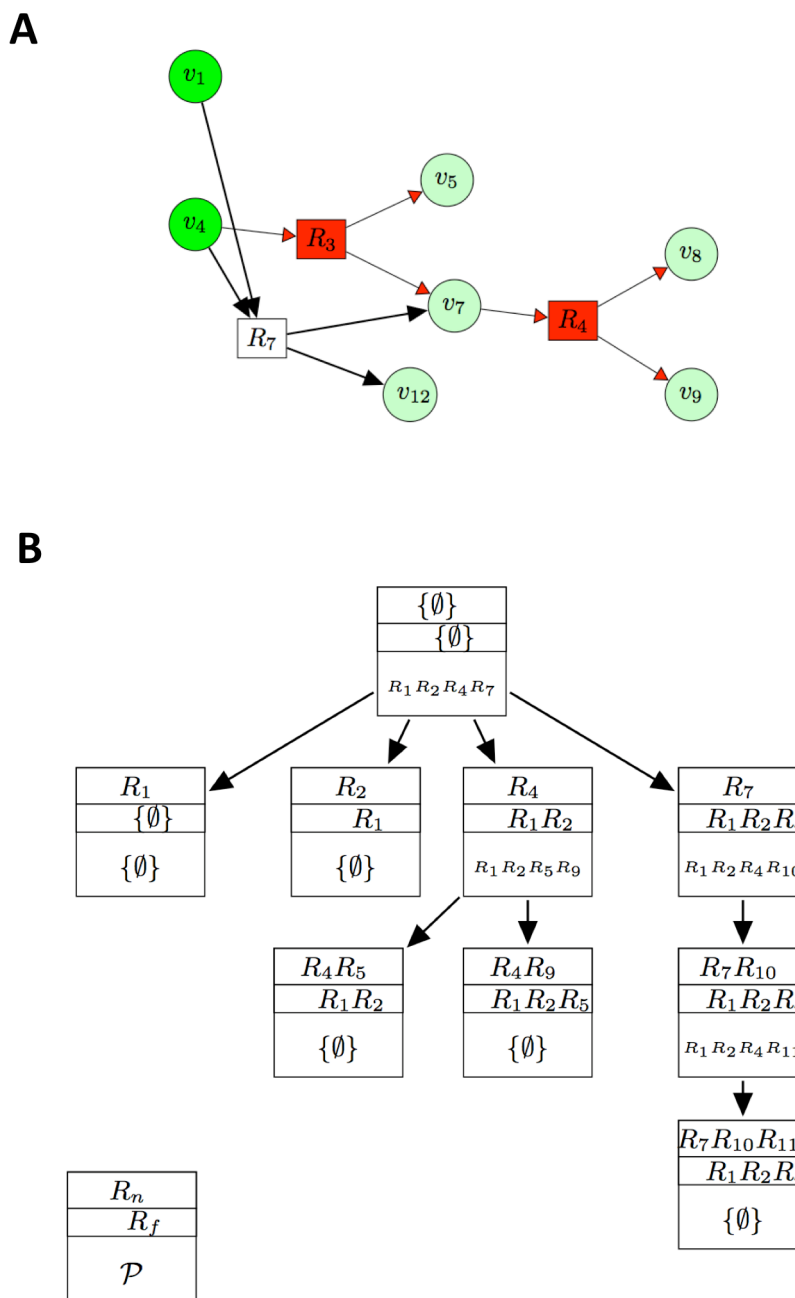


Figure 3 The resolution of the enumeration problem. A) The process of enumeration: The hyperpath given by running *FindAll* on the hypergraph of Figure 2D. This hyperpath is not minimal as can be verified by running *Minimize* without constraints (i.e. $R_f = \{\emptyset\}$). Two minimal hypergraphs connect v_8 to the source nodes v_1, v_4 : the one containing hyperarcs R_4, R_7 and the one highlighted in the figure containing only the hyperarcs R_3, R_4 ; B) Scheme of resolution of the enumeration problem for biosynthesis of liquiritigenin whose hypergraph is represented in Figure 1. Each node in the scheme corresponds to a call to *FindPath* and contains the constraint sets R_f and R_n (see left bottom box), and the minimal hyperpath \mathcal{P} corresponding to the given constraints. *FindPath* iteratively calls itself, the structure obtained is a rooted tree where each node represents a call to *FindPath*, and each call is characterized by the sets R_f and R_n . In particular there are no constraints on the pathway searched at the root node, corresponding to the first call to *FindPath*, this fact is expressed by having empty R_f and R_n . The hyperpath found is R_1, R_2, R_4, R_7 . For each reaction in the hyperpath, a new call to *FindPath* is done, this time with the constraints induced by the hyperpath solution of the parent node. Processes that do not return hyperpaths ($\mathcal{P} = \{\emptyset\}$) are not followed by calls to children processes, while the processes that return one hyperpath \mathcal{P} have unconstrained reactions $\mathcal{P} \setminus R_f$ and are followed by as many children processes as unconstrained reactions. The children processes have as set of reactions that cannot belong to the returned pathway R_n' the same as the father augmented by one reaction from the unconstrained set of the hyperpath \mathcal{P} returned by the father process, while the set of reactions required to belong to the pathway R_f' is given by the ones of the father augmented by all the reactions preceding the one added to R_n to get R_n' . Consider for instance the process giving as pathway $\mathcal{P} = \{R_1, R_2, R_5, R_9\}$ it contains two reactions not in R_f : R_5 and R_9 so its two children processes have R_n given by $R_4 \cup R_5$ and $R_4 \cup R_9$ and R_f is given by the one of the father: $\{R_1, R_2\}$ union once with $\{\emptyset\}$ and once with R_5 : the only unconstrained reaction preceding R_9 in the pathway \mathcal{P} .

```

10.      $En \leftarrow \{En, \text{FindPath}(\mathcal{H} \setminus r, R_f, T, S_{\mathcal{H}})\}$ 
11.      $R_f \leftarrow R_f \cup r$ 
12.     end if
13. end for
14. end if

```

output:

En

The loop at line 7 of *FindPath* is done according to the order given by line 6 where the hyperarcs are ordered so that at least one of the head vertices of each hyperarc is a tail vertex of some previous reaction. Such an ordering is always possible since \mathcal{P} is a hyperpath. As said above and illustrated in Figure 3B, *FindPath* is an algorithm that iteratively calls itself, see line 10. Note that even if R_n is not explicitly defined in *FindPath*, it is constructed implicitly when at line 10 of *FindPath* is called on the smaller graph $\mathcal{H} \setminus r$.

Let us note that the output of the enumeration is not always composed of minimal hyperpaths. This is due to the fact that the algorithm *Minimize* while running in polynomial time can return a non-minimal hyperpath. An algorithm always returning minimal hyperpaths cannot be polynomial since the problem of finding a minimal hyperpath containing a set R_f of hyperarcs is an NP-complete problem as showed in Appendix A.2. However, in many practical instances (for instance when hyperarcs only have one head node), the algorithm *Minimize* returns a minimal constrained hyperpath. As a matter of fact, for all the enumeration studies we have so far carried out, we observed that the output obtained by *Minimize* when called by the algorithm *FindPath* introduced above was always a minimal hyperpath. Nonetheless, a characterization of hard instances of the minimal constrained hyperpath problem is given in Appendix.

Supplements Enumeration

Provided a given metabolic network and a set of source compounds (e.g. a set of compounds in the growth media, a set of endogenous metabolites of a species) it may not be possible to link all the metabolites of the network to the source set. When a target compound is not accessible from the source set, one can consider the possibility of inserting into the metabolism of the organism some precursors so that the target becomes reachable. In practice such a task can be carried out through the enrichment of the growth media. More generally, the insertion of supplements can be used even when the target compound is reachable in order to access to new pathways for the production of the target.

Let a supplement for a target T be any compound $i \notin B(\mathcal{H}, S_{\mathcal{H}})$ that is involved as reactant in at least one minimal hyperpath going from a superset of $S_{\mathcal{H}} \cup i$ to the target T . Below we give the algorithm *FindSupp*

finding the supplements for the production of a given target. In Figure 2D supplements are highlighted in red. Therefore, the process of finding supplements is useful as a general strategy in metabolic engineering in order to determine which metabolites might be part of the metabolism that produces a given target. **Algorithm FindSupp** (Find supplements for the production of the compounds in T , from the source $S_{\mathcal{H}}$ of hypergraph \mathcal{H})

input:

$\mathcal{H}, S_{\mathcal{H}}, T$ (list of compounds to produce)

```

1.  $WishList \leftarrow T$ 
2.  $D \leftarrow \{\emptyset\}$ 
3. while  $WishList \setminus D \neq \{\emptyset\}$ 
4.   let  $i$  be an element of  $WishList \setminus D$ 
5.    $D \leftarrow D \cup i$ 
6.    $Aux \leftarrow \{\emptyset\}$ 
7.   for all reactions  $r$  with  $i \in Y(r)$ 
8.      $Aux \leftarrow Aux \cup (X(r) \setminus (S_{\mathcal{H}} \cup D))$ 
9.   end for
10.   $WishList \leftarrow WishList \cup Aux$ 
11. end while
12.  $F \leftarrow \text{FindAll}(\mathcal{H}, S_{\mathcal{H}})$ 
13.  $D \leftarrow D \setminus S_{\mathcal{H}} \cup (\cup_k Y(F_k))$ 

```

output:

D

Bootstraps

Bootstrap molecules relative to a source $S_{\mathcal{H}}$ are the molecules that cannot be produced by a hyperpath with source $S_{\mathcal{H}}$ unless they are already present in the media. An example of bootstrap nodes are nodes v_2, v_3 of Figure 2D. In this section we give an algorithm finding in polynomial time all the bootstraps of a hypergraph \mathcal{H} with source vertices $S_{\mathcal{H}}$. Bootstraps are special kind of supplement, if at any step of a pathway, a heterologous metabolite is needed as a substrate but has not yet been produced from the source set, then this metabolite is a bootstrap and must be added in the growth media for the reaction to take place, and for the pathway to be a valid pathway. The algorithm given below enables one to detect bootstraps prior enumerating pathways running the *FindPath* algorithm.

Algorithm FindBootstraps (Given a hypergraph \mathcal{H} and a source $S_{\mathcal{H}}$, returns the set B of bootstrap nodes)

input: $\mathcal{H}, S_{\mathcal{H}}$

```

1.  $F \leftarrow \text{FindAll}(\mathcal{H}, S_{\mathcal{H}})$ 
2.  $D \leftarrow S_{\mathcal{H}} \cup (\cup_k Y(F_k))$ 
3.  $\mathcal{H}' \leftarrow \{\emptyset\}$ 
4. for all  $r$  in  $\mathcal{H}$ 
5.    $r' \leftarrow (X(r) \setminus D, Y(r) \setminus D)$ 
6.   if  $Y(r') \neq \{\emptyset\}$ :
7.      $\mathcal{H}' \leftarrow \mathcal{H}' \cup r'$ 
8.   end if

```

```

9. end for
10. while exists  $v$  in  $\cup_{r \in \mathcal{H}} Y(r) \setminus \cup_{r \in \mathcal{H}'} X(r)$ 
11.   for all  $r'$  containing  $v$ :
12.      $r' \leftarrow (X(r'), Y(r') \setminus v)$ 
13.     if  $(Y(r') = \{\emptyset\})$  or  $(v \in X(r'))$ :
14.        $\mathcal{H}' \leftarrow \mathcal{H}' \setminus r'$ 
15.     end if
16.   end for
17. end while
18.  $B = \cup_{r \in \mathcal{H}'} Y(r')$ 
output:
  B

```

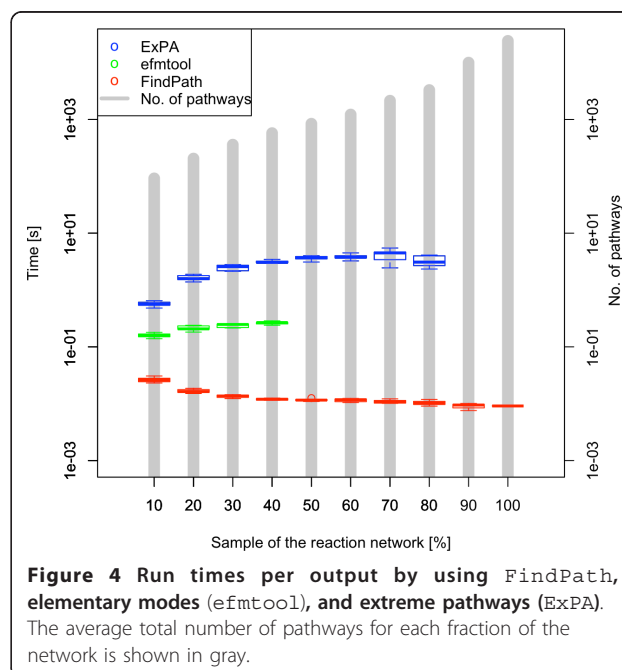
The *FindBootstraps* algorithm is linear in the number of vertices, hyperarcs and in the total coordination. Remark that the set $\cup_{r \in \mathcal{H}'} Y(r')$ obtained in line 18 is equal to $\cup_{r \in \mathcal{H}'} X(r')$. In fact the bootstrap vertices $b(\mathcal{H}, S_{\mathcal{H}})$ constitute the largest set of vertices not reachable from the source and such that each element of the set belongs to the head of at least one reaction whose tail only contains vertices in $B(\mathcal{H}, S_{\mathcal{H}})$ or in $b(\mathcal{H}, S_{\mathcal{H}})$. Notice that the set of bootstrap vertices in a hypergraph \mathcal{H} only depends on the source vertices and does not depend on the target.

Results and Discussion

To evaluate the performance of our topological approach (e.g. algorithm *FindPath*), we have compared running times of this approach with the running times of the steady state approach presented in the Methods section. All our tests were run on a Mac Pro server 2 × 2.66 Ghz Quad-Core Intel Xeon, 16 GB. All the algorithms of the topological approach were implemented in Python. For the steady state approach we used two software products, one computing elementary modes and the other extreme pathways. Elementary modes were computed by using the MATLAB interface to the Java implementation of *efmtool* Version 4.7.1 [7]. Extreme pathways were computed by using the Mac OSX version of the *ExPA* program [6]. The running time comparison test was performed for different random samples of the hypergraph \mathcal{H} constructed from the KEGG database [27]. *E. coli* was chosen as source organism. The hypergraph \mathcal{H} was composed of 6542 metabolites connected by 8392 reactions including 971 metabolites endogenous to *E. coli* and 5571 heterologous compounds. Each sampled hypergraph \mathcal{H}_s was built by randomly sampling a given fraction of the total reactions in the hypergraph \mathcal{H} . Tests for each sampling fraction were repeated 10 times. In the case of elementary modes, we were able to run the test only up to 50% sampling of the full metabolic network, due to memory constraints in MATLAB. For extreme pathways, the test was run up to 85% due again to memory constraints as well. Prior to

enumerating pathways with elementary modes, extreme pathways or the direct algorithm, bootstraps were identified by using the algorithm *FindBootstraps* defined in Methods and added to the growth media.

The observed trend is consistent in the three cases, as it is shown in Figure 4. For the network composed of all 6542 metabolites and 8392 reactions, there are 23564 pathways connecting 2338 out of 5571 heterologous metabolites to the source. This relative low number of pathways producing heterologous compounds is related to the fact that heterologous maps usually show a degree of hierarchy higher than the one observed in native metabolic networks such as central metabolism (a comparison is provided in supplementary Additional file 1, Figure S1). It is indeed not surprising to find the same number of extreme pathways and elementary modes as all reactions in our networks are irreversible. In the case of *FindPath*, the fact that the topological approach does not take stoichiometry into account might produce some inconsistent pathways. Checking that a given pathway is stoichiometrically balanced can be done using linear programming [28]. In the context of metabolic engineering (i.e. enumerating pathways between native metabolites and heterologous targets) we found only few inconsistent pathways (less than 1% of the 23564 enumerated pathways). These pathways are eliminated after enumeration by our ranking function as one of the criteria to rank pathways is to solve the steady state equations [1]. Once unbalanced pathways were removed from the list of enumerated pathways, we obtained the same pathways as through the calculation



of elementary modes. These results are consistent with the given definitions of the algorithms. In a forthcoming paper, we plan to generalize them into a formal proof.

We also find that FindPath has the fastest execution time, the algorithm required less than 5 min to compute the full network. Elementary modes has an execution time approximately 10-fold slower than FindPath. The computation of extreme pathways, in turn, is the slowest one, being between $10^2 \sim 10^3$ -fold slower than FindPath. The execution times are on average of 0.0136 ± 0.0051 s per output with the FindPath algorithm, of 0.218 ± 0.042 s with elementary modes, and of 2.84 ± 1.24 s with extreme pathways. The observed execution times mean that for the compound with the highest number of pathways in our tests, the anti-diabetic drug acarbose containing 1513 pathways, enumerating the pathways with elementary modes takes more than 329.83 seconds, while it can be computed in 20.58 seconds with FindPath. We have measured running times per output and memory usage as function of input size and output size (see Table 1 and Additional file 2, Figure S2). In all cases we found linear growth $O(n)$ for both input size and output size. In the case of running times, FindPath and efmtool have running times per output approximately constant in function of the size of the input stoichiometric matrix S of 3.1×10^{-2} and 1.6×10^{-1} seconds, respectively, being ExPA the less efficient code with a constant time of 4.2×10^{-1} seconds and a linear growth of 1.6×10^{-7} seconds per input. Similar values were obtained depending on the size of the output, although the scaling in this case for ExPA was of 5.7×10^{-3} seconds per output. Regarding memory usage, ExPA and FindPath are less demanding, especially for output size, with linear growths of 3.9 and 54 kB per output, respectively, while efmtool required significantly more memory allocation, 3.3×10^4 kB per output.

To the best of our knowledge, the computational complexity of enumerating elementary modes on networks comprising irreversible reactions is up-to-date unknown [28]. In Appendix A.2 we prove that enumerating minimal pathway is an NP-complete problem (reduction to 3-SAT) but as exemplified in Appendix B

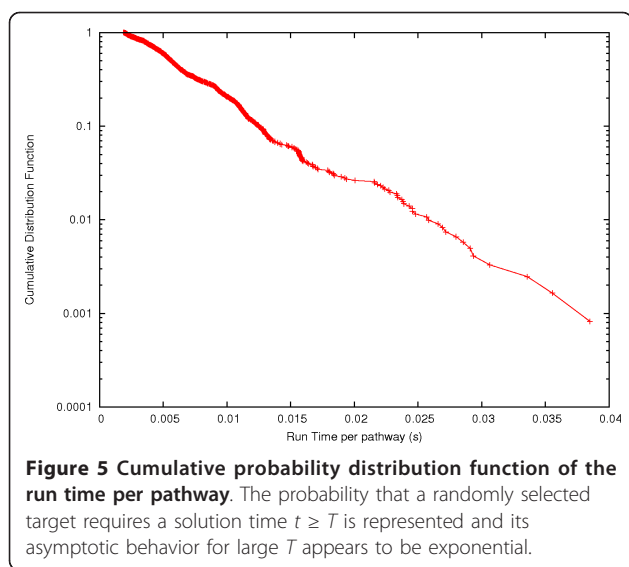
the hard instances are rather rare and obviously none of these hard instances were found in our running tests as all pathways returned by FindPath were indeed minimal pathways.

To further probe the computational complexity of FindPath we computed distribution of the run time for each of the 2338 heterologous targets that are linked to *E. coli*. As shown in Figure 5, the distribution is exponentially distributed, the average run time is thus finite. Examples of pathway enumeration for heterologous compounds with therapeutical applications were provided in our previous methodology study about pathway ranking [1] for penicillin N and taxol (see also Additional file 4, Figure S5 and S6). These two examples contained a relative low level of combinatorial complexity, with a maximum number of 14 different pathways producing penicillin N. The combinatorial complexity of enumerating all putative biosynthetic routes producing a target compound, however, might become considerably higher than in these two previous examples (cf. Figure 2 in [1] where more than 10000 pathways can be found between tyrosine and chorismate). Further examples are those involved in the biosynthetic pathways for plant steroids leading to brassinolide, where 8 pathways can be used to produce campesterol, one of the initial precursors. The number of pathways grows as we proceed downstream going up to 40 for teasterone, and to 224 for typhasterol. Finally, the number of pathways for the end products, castasterone and brassinolide is of 328. This example illustrates how the complexity of pathways enumeration can grow with the number of intermediates involved in the synthesis of the final product. Since FindPath algorithm has been designed with the aim to enumerate pathways in metabolic engineering applications, it does not directly address other general metabolic network analysis problems like finding shortest paths between two compounds of endogenous pathways in central metabolisms. Indeed, FindPath does not enumerate pathways comprising compounds (bootstraps) that are consumed before being produced unless they were added as supplement in the growth

Table 1 Performance comparisons

	Code	Run time per output [s]	Memory use [kB]
Size of input S [$n \times m$]	FindPath	$3.1 \times 10^{-2} - 1.1 \times 10^{-9}x$	$4.2 \times 10^{+1} + 1.5 \times 10^{-2}x$
	efmtool	$1.6 \times 10^{-1} + 6.7 \times 10^{-9}x$	$2.0 \times 10^{+5} + 9.6 \times 10^{-2}x$
	ExPA	$4.2 \times 10^{-1} + 1.6 \times 10^{-7}x$	$3.3 \times 10^{+3} + 1.1 \times 10^{-5}x$
Size of output [no. of pathways]	FindPath	$3.2 \times 10^{-2} - 4.0 \times 10^{-5}x$	$-1.6 \times 10^{+1} + 5.4 \times 10^{+1}x$
	efmtool	$1.5 \times 10^{-1} + 2.3 \times 10^{-4}x$	$1.0 \times 10^{+5} + 3.3 \times 10^{+4}x$
	ExPA	$2.5 \times 10^{-1} + 5.7 \times 10^{-3}x$	$3.2 \times 10^{+3} + 3.9 \times 10^{+0}x$

Performance of the codes FindPath, efmtool and Expa in terms of run time per output and memory use in function of size of input stoichiometric matrix S and size of output (number of pathways).



medium (cf. in Additional file 3 an example of pathway enumeration with FindPath using bootstrap compounds).

Another aspect of pathway design is the definition of a cost function that estimates limiting effects on production efficiency by multiple factors associated with genetic and metabolic engineering of the pathway, such as gene heterogeneity, metabolite toxicity, or steady-state fluxes. In order to facilitate the designer in the selection of the best synthetic pathways to be implemented each hyperpath enumerated by our *FindPath* algorithm is ranked depending on this cost estimation. Once a cost function to minimize is introduced, the search for an optimal pathway can be formulated as a shortest path problem. The shortest path problem for weighted graphs consists of finding a path going from a given source vertex to a given target vertex while minimizing a cost function given by the sum of the costs of the arcs involved in the path. In order to define a shortest-hyperpath problem we need a definition of cost for hyperpaths in a weighted hypergraph (i.e. a hypergraph whose hyperarcs have associated a non-negative real number representing their cost). A natural generalization of the cost function for paths in graphs to hypergraphs is the sum of the costs of the hyperarcs contained in the hyperpath. If on the one hand this generalization seems to be natural, on the other hand the two problems have, however, not the same complexity. In fact the shortest-hyperpath problem with this cost function is known to be an NP-hard problem (see Appendix A.1).

The reason why the algorithms commonly used for graphs cannot be easily adapted for hypergraphs is that given the costs $W(\mathcal{P}, a_i)$ for the hyperpaths $A_i \subset \mathcal{P}$ leading to the vertices a_{i_s} tails of a hyperarc e_j whose

cost is w_j , then the cost of the hyperpath containing all the hyperarcs in $\cup_i A_i$ and e_j is not equal to $w_j + \sum_i W(\mathcal{P}, a_i)$ if the overlap of the hyperpaths A_i is non trivial.

Provided an additive cost function as defined in [23], finding the shortest hyperpath can be done in polynomial time by an algorithm of complexity $O(m \cdot (n + \ln m))$ given in [23], which finds the shortest hyperpath in B -hyperpaths. In fact, this algorithm does apply to minimal hyperpaths given in definition (4).

Generally speaking, the strategy we used in the enumeration algorithm (iterative partition of the space of feasible solutions) is often used to solve the problem of finding the first k solutions of a combinatorial problem, but it cannot be extended to the k -shortest problem on hypergraphs not even with an additive cost function (cf. definition in Appendix A.1), since when splitting the problem into a constrained part and a free one (by using the sets R_n and R_f) we get a problem that is known to be NP-hard (see the proof in [29] where an algorithm is given for the k -shortest hyperpath on hypergraphs whose hyperarcs have only one head vertex).

Therefore, the k -shortest path problem might be solvable efficiently if one is able to develop a specific cost function making the problem tractable. However such an artifact cost function may not be necessarily appropriate to practical problems, such as the production of heterologous targets using metabolic engineering. As an alternative we have proposed in this paper an enumeration algorithm, which systematically enumerates all pathways linking source compounds to target compounds. In all the cases that we have so far processed for metabolic networks, our algorithm runs in polynomial time per output, and therefore pathways can be ranked by the designer based on their own user-defined cost functions.

Conclusions

In summary, the methods presented in this paper provide metabolic engineers with powerful tools that extend the toolbox for heterologous biosynthetic pathway design. Besides pathway enumeration of biosynthetic routes for a given target product, our methods have several other possible applications. For instance, they can be used in combination with gene deletion strategies in order to determine pathway manipulations leading to overproduction of the target compound. Another application is in biodegradation and bioremediation, where our algorithms would need to be slightly modified in order to reverse the pathway search so that it can identify degradation routes for a given compound, while the underlying structure of the algorithms remains still

valid. Finally beyond metabolism, our algorithms could also be utilized in the context of chemical synthesis to enumerate all the possible routes linking a target molecule to a source set of starting reactants, enabling the search for the best routes in terms of production costs.

Availability and requirements

A web server is available: <http://bioretrosynth.issb.genopole.fr/tools/metahype> See details in Additional file 4.

Appendix A Reduction proofs for the shortest hyperpath problem and the minimal constrained hyperpath problem

A.1 Shortest hyperpath Problem

In [30] a reduction of the shortest-hyperpath problem to Minimum Set Cover (MC) is given. We have to adapt the proof to our case for two reasons: the definition of directed hypergraph that was used is more restrictive (they only admit hyperarcs e_i such that $|Y(e_i)| = 1$) and the hyperpath was ill-defined. In fact the given definition by these authors does not permit to say if some hypergraphs (as the one in Figure 2C) are also hyperpaths or not. In other words their definition is ambiguous: does not permit to determine the nature of all the directed hypergraphs and thus can be completed in several ways.

Nonetheless the proof of NP-hardness they gave is valid for our more general hypergraphs and minimal hyperpaths because the set of directed hypergraphs employed in the reduction proof in [S1] is a sub-ensemble of the directed hypergraphs we defined above in the Definitions section and all the hypergraphs employed for the reduction are well defined as hyperpaths, independently of the way the incomplete definition they gave is completed. Since our definition is a way to complete the definition in [S1], then the two definitions agree on the set of hypergraphs employed for the reduction.

From these facts follows that the shortest hyperpath problem is an NP-hard problem. And, in particular, if the weights on hyperarc are non-negative, since hyperpaths always contain at least one minimal hyperpath, the shortest minimal hyperpath problem is NP-hard too.

Additive cost functions

The reason why the algorithms commonly used for the shortest path problem on graphs cannot be easily adapted for hypergraphs is that given the costs $W(\mathcal{P}, a_i)$ for the hyperpaths $A_i \subset \mathcal{P}$ leading to the vertices a_i , tails of a hyperarc e_j whose cost is w_j , then the cost of the hyperpath containing all the hyperarcs in $\cup_i A_i$ and e_j is not equal to $w_j + \sum_i W(\mathcal{P}, a_i)$ if the overlap of the hyperpaths A_i is non trivial.

In order to define a shortest path problem that can be solved polynomially by a variant of Dijkstra algorithm,

the additive cost functions have been introduced in [23] for the B -hyperpaths. We adapt below the notion of "additive" cost function for hyperpaths. A cost function $W(\mathcal{P}, x)$ returning the cost for reaching the vertex x with the hyperpath \mathcal{P} starting from a source S whose elements $s \in S$ have $W(s) := 0$ is additive if $W(x)$ is the minimum over all the arcs $e_x \in \mathcal{P}$ whose head contains x of $e_x + f(W_i)$ where $W_i := W(\mathcal{P}, i)$ are the costs for reaching the tail vertices i of e_x in \mathcal{P} , f is an increasing monotone function of its argument and $f(W_i) \geq W_i \forall i$. Remark that the cost of a hyperpath determined with an additive cost function in general is not given by the sum of the costs of the hyperarcs.

A.2 Minimal Constrained Hyperpath Problem

Consider a 3-SAT instance concerning n variables σ_i and m clauses X_j consisting of the problem of deciding if there exists an assignment of Boolean values to the σ_i such that all the clauses are satisfied. For each boolean variable σ_i contained in at least one clause introduce one hyperarc ε_i with the head of each ε_i having two vertices $Y(\varepsilon_i) = \{v_{i+}, v_{i-}\}$. For each clause X_j consider a vertex v_j and seven hyperarcs (each one corresponding to boolean assignment of the three variables satisfying the clause X_j). A boolean assignment is a triple a_1, a_2, a_3 of boolean values. Let these hyperarcs be $\mu_{j1}, \dots, \mu_{j7}$ and let the head $Y(\mu_{jk})$ of a hyperarc μ_{jk} corresponding to the combination a_1, a_2, a_3 of the boolean variables $\sigma_{j1}, \sigma_{j2}, \sigma_{j3}$ be $Y(\mu_{jk}) = \{v_{j1a1}, v_{j2a2}, v_{j3a3}, v_j\}$. Now let the tails of each hyperarc introduced be connected to the source nodes. And let consider a node T being the product of the reaction R having as substrates the vertices v_j and the heads $\{v_{i+}, v_{i-}\}$ of the hyperarcs ε_i .

Given the hypergraph described above (whose size is linear in the size of the underlying 3-SAT problem) consider the minimal constrained hyperpath problem where all the hyperarcs ε_i are mandatory, and the target is T .

A solution of this problem gives in linear time a solution for the underlying SAT problem, which makes the problem of minimal constrained hyperpath an NP-complete problem. In fact, given a minimal hyperpath M , solution of this problem, for each i consider v_i^* the only one of the two head vertices $\{v_{i+}, v_{i-}\}$ belonging to the head of one or more of the μ arc in \mathcal{M} (only one of the two vertices can belong to the head of a μ hyperarc in \mathcal{M} because otherwise the hyperarc ε_i would be superfluous). The boolean assignments $\sigma_i = v_i^*$ are a solution of the 3-SAT problem and, inversely if a solution of the 3-SAT problem exists then a minimal pathway satisfying the constraints exists and is the one obtained using only one of the hyperarcs for each X_j among the ones whose head only contains v_j and vertices v_i^* .

In Additional file 5, Figure S8 for simplicity we consider the reduction of a single-clause satisfaction problem to finding if a minimal hyperpath satisfying the constraints exists. There exist seven minimal hyperpaths connecting the target vertices to the source and satisfying the constraint that $\varepsilon_1, \varepsilon_2, \varepsilon_3$ are parts of the hyperpath. Each solution corresponds to a valid boolean assignment of the variables $\sigma_1, \sigma_2, \sigma_3$.

Appendix B Hard instances of minimal constrained hyperpath problem

On many hypergraphs the algorithm enumerating the pathways only returns minimal hyperpaths, this is the case for the metabolic networks that we analyzed in the main sections of this paper. In this section we give a characterization of the hypergraphs where the algorithm *Minimize* solves the minimal constrained hyperpath problem, characterizing these instances helps to individuate which hypergraphs are expected to give an output only containing minimal hyperpaths.

Let $Y(R_f)$ be the set of all the metabolites produced by reactions in R_f , the mandatory reactions: $Y(R_f) := \cup_{r \in R_f} Y(r)$. Given a hypergraph \mathcal{H} and the sets R_f, R_n we say that the well-separation condition holds if for every reaction $r \in \mathcal{H} \setminus (R_f \cup R_n)$ the set $Y(r)$ of products of r either is a subset of $Y(R_f)$ or does not contain elements of $Y(R_f)$. If the well-separation condition holds for a hypergraph \mathcal{H} with constrained reactions R_f , the algorithm *Minimize* returns a minimal hyperpath solving the minimal constrained hyperpath problem if a solution exists.

The well-separation condition holds for every choice of R_f in a hypergraph whose reactions have one only product, as the hypergraphs defined in [29]. If on the one hand, this condition can appear too constraining, on the other hand it can be generalized to larger sets of hypergraphs. For instance, the algorithm *Minimize* returns a minimal hyperpath solving the minimal constrained hyperpath problem even if the well-separation condition holds on the pruned graphs obtained by keeping from the original hypergraph only the reactions belonging at least to one hyperpath linking the target to the source and only the nodes being tail of such reactions.

Examples of hard instances can be found among the ones used for the proof of NP-completeness. In general, hard instances \mathcal{H} of the enumeration problem have to violate the condition of well-separation for some choice of $R_f \subset \mathcal{H}$, in order that the corresponding minimal constrained hyperpath problem becomes hard. This happens if several compounds are products of more than one reaction producing more than one compound.

This is the case for nested networks as the one in Additional file 6, Figure S9. While the given network is small enough to be solvable by hand, it contains nevertheless the principal ingredients of complexity that would asymptotically make harder the problem as the size of the instances grows.

Finding one minimal hyperpaths leading to the production of v_8 is a simple problem, but finding new ones gets more and more involved. This is a consequence of the fact that the nodes v_1, v_2, v_3, v_4, v_5 can be produced by different choices of the reactions R_1, R_2, R_3, R_4, R_5 and each of these reaction has more of one product susceptible to participate to the production of the target.

Additional material

Additional file 1: Figure S1. Distribution of graph hierarchies (Butts, C J *Stat Soft*, 24:1-50, 2008) in heterologous metabolic networks (0.169 \pm 0.11) in comparison with the graph hierarchy of central (0.032), nucleotide (0.027), lipid (0.051), and amino acid (0.030) metabolic networks in *E. coli*.

Additional file 2: Figure S2. Performance comparisons for `FindPath`, `ExPA` and `efmtools` for run time per output and memory use in function of size of input and size of output.

Additional file 3: An example of pathway enumeration with bootstraps. An example of pathways enumeration with `Findpath` using bootstraps.

Additional file 4: Availability and requirements. Description of the available MetaHype web server for running the algorithms for KEGG compounds.

Additional file 5: Figure S8. Minimal constrained hyperpath problem reduction of a 3-SAT formula.

Additional file 6: Figure S9. Instance of minimal constrained hyperpath problem.

Acknowledgements

Funding: Genopole[®] through an ATIGE grant; ANR through a Chair of Excellence.

Authors' contributions

JLF and PC designed the algorithms and the experiments. DF implemented the algorithms, and wrote the proof of the NP-completeness for the minimal constrained hyperpath problem. SBP wrote the example of pathway enumeration. PC implemented the web service prototype, performed the experiments and analyzed the results. All authors wrote the manuscript. All authors read and approved the final manuscript.

Received: 2 August 2011 Accepted: 6 February 2012

Published: 6 February 2012

References

1. Carbonell P, Planson AG, Fichera D, Faulon JL: A retrosynthetic biology approach to metabolic pathway design for therapeutic production. *BMC Systems Biology* 2011, **5**:122+.
2. Planson AG, Carbonell P, Paillard E, Pollet N, Faulon JL: Compound toxicity screening and structure-activity relationship modeling in *Escherichia coli*. *Biotechnol Bioeng* 2011, **109**(3):846-850.
3. Edwards J, Ramakrishna R, Schilling C, Palsson B: *Metabolic Engineering*, Marcel Dekker 1999, 13-57.
4. Fell D: *Metabolic Networks*. *Complex Systems and Interdisciplinary Science World Scientific* 2007, **3**:163-198.

5. Llaneras F, Picó J: Which metabolic pathways generate and characterize the flux space? A comparison among elementary modes, extreme pathways and minimal generators. *J Biomed Biotechnol* 2010, **2010**.
6. Bell SL, Palsson B: expa: a program for calculating extreme pathways in biochemical reaction networks. *Bioinformatics* 2005, **21**(8):1739-1740.
7. Terzer M, Stelling J: Large-scale computation of elementary flux modes with bit pattern trees. *Bioinformatics* 2008, **24**(19):2229-2235.
8. Motzkin TS, Raiffa H, Thompson GL, Thrall RM: The double description method. In *Contributions to the theory of games. Volume 2*. Princeton, NJ: Princeton University Press; 1953:51-73.
9. Pharkya P, Burgard AP, Maranas CD: OptStrain: A computational framework for redesign of microbial production systems. *Genome Res* 2004, **14**(11):2367-2376.
10. de Figueiredo LF, Podhorski A, Rubio A, Kaleta C, Beasley JE, Schuster S, Planes FJ: Computing the shortest elementary flux modes in genome-scale metabolic networks. *Bioinformatics* 2009, **25**(23):3158-3165.
11. Pey J, Prada J, Beasley J, Planes F: Path finding methods accounting for stoichiometry in metabolic networks. *Genome Biol* 2011, **12**(5):R49+.
12. Hatzimanikatis V, Li C, Ionita JA, Henry CS, Jankowski MD, Broadbelt LJ: Exploring the diversity of complex metabolic networks. *Bioinformatics* 2005, **21**(8):1603-1609.
13. Cho A, Yun H, Park JHH, Lee SYY, Park S: Prediction of novel synthetic pathways for the production of desired chemicals. *BMC Systems Biology* 2010, **4**:35+.
14. Croes D, Couche F, Wodak SJ, van Helden J: Metabolic PathFinding: inferring relevant pathways in biochemical networks. *Nucleic Acids Res* 2005, **33**(suppl 2):W326-W330.
15. Croes D, Couche F, Wodak SJ, van Helden J: Inferring meaningful pathways in weighted metabolic networks. *J Mol Biol* 2006, **356**:222-236.
16. Faust K, Croes D, van Helden J: Metabolic Pathfinding Using RPAIR Annotation. *J Mol Biol* 2009, **388**(2):390-414.
17. Rahman SA, Advani P, Schunk R, Schrader R, Schomburg D: Metabolic pathway analysis web service (Pathway Hunter Tool at CUBIC). *Bioinformatics* 2005, **21**(7):1189-1193.
18. Kotera M, Hattori M, Oh MA, Yamamoto R, Komeno T, Yabuzaki J, Tonomura K, Goto S, Kanehisa M: RPAIR: a reactant-pair database representing chemical changes in enzymatic reactions. *Genome Informatics* 2004, **15**:P062+.
19. Kotera M, Okuno Y, Hattori M, Goto S, Kanehisa M: Computational assignment of the EC numbers for genomic-scale analysis of enzymatic reactions. *J Am Chem Soc* 2004, **126**(50):16487-16498.
20. Romero P, Karp P: Nutrient-related analysis of pathway genome database. *Pac Symp Biocomput* 2001, 471-482.
21. Cottret L, Milreu P, Acuna V, Marchetti-Spaccamela A, Viduani-Martinez F, Sagot M, Stougie L: Enumerating precursor sets of target metabolites in a metabolic network. *Lecture Notes in Bioinformatics, Volume LNBI 5251* Springer-Verlag; 2008, 233-244.
22. Nielsen LR, Andersen KA, Pretolani D: Finding the K shortest hyperpaths. *Comput Oper Res* 2005, **32**(6):1477-1497.
23. Gallo G, Longo G, Nguyen S, Pallottino S: Directed Hypergraphs and Applications. *Discrete Appl Math* 1993, **42**:177-201.
24. Klamt S, Gagneur J, von Kamp A: Algorithmic approaches for computing elementary modes in large biochemical reaction networks. *IEE Proc Systems Biology* 2005, **152**(4):249-255.
25. Jevremovic D, Boley D, Sosa CP: Divide-and-Conquer Approach to the Parallel Computation of Elementary Flux Modes in Metabolic Networks. *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, IEEE 2011* 502-511.
26. Schuster S, Fell D, Dandekar T: A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic network. *Nat Biotechnol* 2000, **18**:326-332.
27. Kanehisa M, Goto S, Furumichi M, Tanabe M, Hirakawa M: KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Res* 2010, **38**:D355-D360.
28. Acuna V, Chierichetti F, Lacroix V, Marchetti-Spaccamela A, Sagot M, Stougie L: Modes and cuts in metabolic networks: Complexity and algorithms. *BioSystems* 2009, **95**:51-60.
29. Nielsen L, Andersen K, Pretolani D: Finding theK shortest hyperpaths. *Computers & Operations Research* 2005, **32**:1477-1497.
30. Ausiello G, Italiano G, Nanni U: *Optimal traversal of directed hypergraphs* Berkeley, CA: International Computer Science Institute; 1992.

doi:10.1186/1752-0509-6-10

Cite this article as: Carbonell et al.: Enumerating metabolic pathways for the production of heterologous target chemicals in chassis organisms. *BMC Systems Biology* 2012 **6**:10.

Submit your next manuscript to BioMed Central and take full advantage of:

- Convenient online submission
- Thorough peer review
- No space constraints or color figure charges
- Immediate publication on acceptance
- Inclusion in PubMed, CAS, Scopus and Google Scholar
- Research which is freely available for redistribution

Submit your manuscript at
www.biomedcentral.com/submit

