Computing Robust Optimal Factories in Metabolic Reaction Networks

Spencer Krieger[1(✉)] and John Kececioglu[2]

[1] Ray and Stephanie Lane Computational Biology Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
skrieger@andrew.cmu.edu

[2] Department of Computer Science, The University of Arizona, Tucson, AZ 85721, USA
kece@cs.arizona.edu

Abstract. Perhaps the most fundamental model in synthetic and systems biology for inferring pathways in metabolic reaction networks is a metabolic *factory*: a system of reactions that starts from a set of source compounds and produces a set of target molecules, while conserving or not depleting intermediate metabolites. Finding a shortest factory—that minimizes a sum of real-valued weights on its reactions to infer the most likely pathway—is NP-complete. The current state-of-the-art for shortest factories solves a mixed-integer linear program with a major drawback: it requires the user to set a critical parameter, where too large a value can make optimal solutions infeasible, while too small a value can yield degenerate solutions due to numerical error.

We present the first *robust algorithm* for optimal factories that is both *parameter-free* (relieving the user from determining a parameter setting) and *degeneracy-free* (guaranteeing it finds an optimal nondegenerate solution). We also give for the first time a *complete characterization* of the graph-theoretic structure of shortest factories via cuts of *hypergraphs* that reveals two important classes of degenerate solutions which were overlooked and potentially output by the prior state-of-the-art. In addition we settle the relationship between the two established pathway models of *hyperpaths* and factories by proving that hyperpaths are actually a subclass of factories. Comprehensive experiments over all instances from the standard metabolic reaction databases in the literature demonstrate our algorithm is *fast in practice*, quickly finding optimal factories in large real-world networks containing thousands of reactions.

A preliminary implementation of our algorithm for robust optimal factories in a new tool called `Freeia` is available free for research use at http://freeia.cs.arizona.edu.

1 Introduction

Metabolic pathways are cornerstones of synthetic and systems biology. They inform metabolic engineering, govern cellular environmental response, and their perturbation has been implicated in the cause of disease [19]. Reactions in such

J. Ma (Ed.): RECOMB 2024, LNCS 14758, pp. 253–269, 2024.
https://doi.org/10.1007/978-1-0716-3989-4_16

pathways are typically annotated with stoichiometry ratios for their participating molecules, that specify the relative number of copies that are consumed and produced. Networks of metabolic pathways are traditionally represented using conventional graphs [23,24], though such graphs do not accurately model multiway reactions that have multiple reactants and multiple products [8,22]. Directed *hypergraphs* (a generalization of directed graphs) in contrast fully capture multiway reactions [8], and represent such a reaction by a single hyperedge, directed from its set of input reactants to its set of output products.

A fundamental task on metabolic networks is to infer the most likely pathway that produces specific target molecules from the source compounds available to the cell, while not exhausting intermediate metabolites. Computationally this corresponds to the *shortest factory* problem we consider here: Given a metabolic network represented by a directed hypergraph with annotated stoichiometries, and a set of sources and targets, find a metabolic factory (a collection of reactions that conserve or do not deplete intermediate metabolites) that produces all the targets from the sources, while minimizing a weighted sum over its reactions.

Next we briefly summarize related work, and then state our contributions.

Related Work. The two main hypergraph models for pathway inference in metabolic networks are hyperpaths and factories. (For a further review, see [9].)

Hyperpaths informally are a set of reactions that produces all targets from the sources, where the reactions can be ordered so that the inputs to each reaction are produced as outputs of preceding reactions. Italiano and Nanni [6] proved that finding shortest hyperpaths is NP-complete. Ritz et al. [20,21] gave the first practical exact algorithm for *acyclic* shortest hyperpaths. Krieger and Kececioglu developed the first methods for *general* shortest hyperpaths that allow cycles: both an efficient heuristic that is close to optimal in practice [10,13,14], and a practical exact algorithm that uses a cutting-plane approach [15–17].

Factories informally are a set of reactions that produces all targets from the sources, while conserving or not depleting intermediate metabolites. Factories have been studied in the context of *min-source* factories (which use the fewest possible source compounds), and *min-edge* factories (which use the fewest possible reactions). Cottret et al. [4] proved that finding a min-source factory is NP-complete, while Acuña et al. [1] and Andrade et al. [2] developed methods to enumerate all min-source factories. Krieger and Kececioglu [12] proved that finding a min-edge factory is NP-complete, and developed a practical exact algorithm that finds optimal factories while also incorporating negative regulation. All these methods (whether explicitly or implicitly) rely on specifying the value of a critical *parameter*, whose default value can exclude valid factories. Furthermore these methods may return *degenerate* solutions that are either non-physical, or correspond to an equilibrium state where targets do not actually accrue.

We will unify these two pathway models—hyperpaths and factories—by later proving that hyperpaths are in fact a *subclass* of factories.

Our Contributions. In contrast to prior work, we give the first robust method for optimal factories that never fails due to poor parameter choices, and always delivers nondegenerate solutions. We also resolve the relationship between hyper-

paths and factories via a new structural characterization of shortest factories. More specifically, we make the following contributions.

- We develop the first *robust algorithm* for shortest factories with no user-specified parameters, whose solution is guaranteed to be nondegenerate.
- We derive the first *complete characterization* of the graph-theoretic structure of reactions in shortest factories in terms of crossing hypergraph cuts.
- We leverage this characterization to unify the main hypergraph models for pathway inference, showing hyperpaths are actually a *subclass* of factories.
- Our computational results reveal the current *state-of-the-art fails* to find factories when they exist, due to brittleness in default parameter settings.
- Our new algorithm is *fast in practice*, as comprehensively demonstrated on the standard pathway databases, with a median runtime under 5 seconds.

A preliminary implementation of our robust algorithm for optimal factories in a new tool called `Freeia` (short for "robust optimal factories in metabolic reaction networks") is freely available at http://freeia.cs.arizona.edu.

Plan of the Paper. The next section presents our new *parameter-free* algorithm for optimal factories. Section 3 gives a complete *characterization* of the structure of shortest factories, proves hyperpaths comprise a subclass of factories, and develops an algorithm for optimal *nondegenerate* factories. Section 4 evaluates our algorithm on real biological benchmarks, highlighting instances where factories were missed by the prior state-of-the-art. Finally Sect. 5 concludes.

2 Computing Optimal Parameter-Free Factories

We next provide the necessary background on factories and hypergraphs, and then present our parameter-free algorithm for shortest factories.

2.1 Factories and Hypergraphs

Informally, a *factory* in a metabolic network is a collection of reactions that produces a set of target molecules starting from a set of source compounds, properly taking into account the stoichiometries of intermediate metabolites in reactions. The reactions in the factory may form cycles, and effectively can proceed simultaneously. This is in contrast to the notion of a *hyperpath*, which is also a collection of reactions that produces the targets from the sources, but without taking into account stoichiometry, and whose reactions must have an ordering in which for each successive reaction all its input reactants are formed as output products of prior reactions in the ordering.

For the intermediate metabolites involved in a factory (the substances other than sources and targets), the stoichiometry ratios for the input reactants and output products of the factory's reactions must be such that one of two conditions are met: either intermediate metabolites neither build up nor get depleted as the

factory continues to produce the targets, known as *conservation*; or intermediate metabolites are allowed to build up, but not be depleted, known as *accumulation*. Under conservation or accumulation, by continuously supplying just the source compounds to the factory, all targets will be produced indefinitely.

To properly represent the reactions in a metabolic network, where a given reaction can have multiple input reactants and multiple output products, requires a generalization of ordinary directed graphs known as a directed *hypergraph* $G = (V, E)$, consisting of a set of directed *hyperedges* E, corresponding to the reactions of the network, and a set of vertices V, corresponding to the substances participating in the reactions. Each hyperedge $e \in E$ is an ordered pair (X, Y) where both $X, Y \subseteq V$ are nonempty sets of vertices, and e is directed from set X to set Y. Here X is called the *tail* of e, and Y is called its *head*, given by functions $\text{tail}(e) = X$ and $\text{head}(e) = Y$. We refer to the in-edges of a vertex $v \in V$ by $\text{in}(v) = \{e \in E : v \in \text{head}(e)\}$, and its out-edges by $\text{out}(v) = \{e \in E : v \in \text{tail}(e)\}$.

For a reaction represented in hypergraph G by hyperedge e, the set $\text{tail}(e)$ is all its input reactants, while $\text{head}(e)$ is all its output products. For a reversible reaction, we represent it in G by a pair of hyperedges $e = (X, Y)$ and its reverse hyperedge $\text{rev}(e) = (Y, X)$. Typically for a metabolic network represented by hypergraph G, the *sources* $S \subseteq V$ of the network are vertices with no in-edges, while *targets* $T \subseteq V$ are often (but not always) vertices with no out-edges.

Figure 2 in Sect. 4.2 shows a directed hypergraph.

Key to metabolic factories is the notion of *flux*: the relative rate at which each reaction is used in its forward direction by the factory. In a hypergraph, we represent the flux for a factory by a nonnegative real-valued vector $f = (f_e)_{e \in E}$ with all $f_e \geq 0$. For a metabolic network represented by a hypergraph, the stoichiometry ratios of the substances in the reactions of the network can be summarized by *stoichiometry matrix* $M = (r_{ve})_{v \in V, e \in E}$ where r_{ve} is the stoichiometry ratio for substance v in reaction e. We express this quantity as $r_{ve} := r^+_{ve} - r^-_{ve}$, with r^+_{ve} being the nonnegative stoichiometry ratio for $v \in \text{head}(e)$, where v is produced as an output product of reaction e; and r^-_{ve} being the nonnegative ratio for $v \in \text{tail}(e)$, where v is consumed as an input reactant of e. In all other cases, r^+_{ve} and r^-_{ve} are zero. Their net difference in r_{ve} is positive when e produces more v than it consumes, and negative in the opposite situation.

Using stoichiometry matrix M, we can express whether flux f for a factory satisfies conservation or accumulation. For a set $I \subseteq V$ of intermediate metabolites, denote by $M\big|_I$ matrix M restricted to rows in I. Then matrix-vector product $M\big|_I f$ is a vector giving for each intermediate metabolite $v \in I$ the relative excess of v produced by reactions in the factory under flux f. Condition $M\big|_I f = 0$ captures *conservation*, while $M\big|_I f \geq 0$ captures *accumulation*.

A hyperedge e with nonzero flux $f_e > 0$ is called an *active edge*, meaning its corresponding reaction is used by the factory.

We now formally define the problem of finding an optimal factory in a metabolic network. This problem has two versions, according to whether we require conservation or accumulation of intermediate metabolites.

Definition 1 (Shortest Factory). *The* Shortest Factory *problem is as follows.*

The input is a metabolic network represented by hypergraph $G = (V, E)$ *with stoichiometries* M*, sources* $S \subseteq V$*, targets* $T \subseteq V - S$*, and edge weight function* ω*.*

The output is nonnegative flux f *such that: for all intermediate metabolites* $I = V - (S \cup T)$*, either accumulation* $M\big|_I f \geq 0$*, or conservation* $M\big|_I f = 0$ *holds; for each target* $t \in T$*, the production requirement* $\sum_{e \in \text{in}(t)} r_{te} f_e > 0$ *is met; and the total weight of active hyperedges* $\sum_{e \in E\,:\, f_e > 0} \omega(e)$ *is minimum.* □

This finds a metabolic factory, given by flux f, that produces all targets T from the sources S, while minimizing the total weight of its active edges. We call this total active edge weight the *length* of the factory.

Figure 2 in Sect. 4.2 illustrates an optimal factory as well.

For *nonuniform* edge weights, under a simple likelihood model where $P(e)$ is the probability that reaction e occurs given its input reactants, and assuming independence of reactions, a shortest factory for weights $\omega(e) = -\log P(e)$ corresponds to a metabolic factory of *maximum likelihood*.

For *uniform* edge weights, the *Minimum-Hyperedge Factory* problem is Shortest Factory with unit weights $\omega(e) = 1$ for all $e \in E$. This finds a factory that uses the least number of reactions, corresponding to a factory of *maximum parsimony*. We call an optimal solution to this problem a *min-edge factory*.

Shortest Factory—even for unit edge weights—is NP-complete [12], so there is likely no algorithm that finds optimal factories and is worst-case efficient.

We note that factories satisfying conservation tend to exist far less frequently than factories satisfying accumulation, as shown in Sect. 4.1 through Table 2.

2.2 Parameter-Free Shortest Factories

We now give the first parameter-free algorithm for Shortest Factory, which solves mixed-integer linear programs (MILPs). We first review the MILP for the state-of-the-art *parameter-based* algorithm, then present the MILPs for our *parameter-free* algorithm for min-edge factories, and finally extend it to *weighted* factories.

Parameter-Based Algorithm. Given an instance of Shortest Factory, the current state-of-the-art parameter-based algorithm [12] constructs an MILP consisting of variables, constraints, and an objective function, as we review below.

The *variables* are grouped into flux vector $f = (f_e)_{e \in E}$ with real-valued variables f_e, and active-edge vector $x = (x_e)_{e \in E}$ with integer-valued variables x_e.

The basic *constraints* are in the following classes. The domain constraints are $0 \leq f_e \leq 1$ and $0 \leq x_e \leq 1$ for all hyperedges $e \in E$ (which ensures $x_e \in \{0, 1\}$). For the intermediate metabolites $I = V - (S \cup T)$, we have either the conservation constraints $M\big|_I f = 0$, or the accumulation constraints $M\big|_I f \geq 0$. For hyperedges $e \in E$, the active edge constraints $x_e \geq f_e$ ensure $x_e = 1$ for an active edge e with $f_e > 0$. Lastly for pairs of reverse hyperedges e and $\text{rev}(e)$ that model a single reversible reaction, the reversible-reaction constraints $x_e + x_{\text{rev}(e)} \leq 1$ prevent trivial cycles that send flux through both e and its reverse.

The *objective function* minimizes $\sum_{e \in E} \omega(e)\, x_e$, the total active edge weight.

All prior methods for optimal factories [2,4,12,25] are *parameter-based*—which explicitly or implicitly depend on a parameter that either must be set by the user or is hard-coded into their implementation—and effectively use a target production constraint for each target $t \in T$ of the form $\sum_{e \in \text{in}(t)} r_{te} f_e \geq \epsilon$, for a small target production constant $\epsilon > 0$, to ensure their solution produces all targets. (This nonzero ϵ is necessary because MILPs cannot accurately represent strict inequalities of the form $\sum_e r_{te} f_e > 0$.) Such a dependence on *parameter* ϵ is a serious shortcoming, as an ill-suited value for ϵ that is too large can prevent the MILP solver from finding an optimal factory (when no optimal factory produces that much of its targets with bounded fluxes), while a value too small can cause the MILP solver to return an invalid factory (due to numerical rounding error).

Parameter-Free Algorithm. Instead our parameter-free algorithm overcomes this through the following approach, using one additional variable, two additional types of constraints, and now solving a small *series* of MILPs. We add a real-valued variable δ, and for each target $t \in T$ we replace the target production constraint by $\sum_{e \in \text{in}(t)} r_{te} f_e \geq \delta$. Then for Minimum-Hyperedge Factory, we add a length constraint $\sum_{e \in E} x_e \leq \ell$, for an integer ℓ that is determined below.

The new objective function now maximizes the value of variable δ.

Notice the objective function value δ^* of an optimal solution to this new MILP gives the maximum possible flux that can be sent to all the targets by a factory with at most ℓ active hyperedges. If δ^* is zero, we know there is no factory of length at most ℓ that produces all the targets. Once we find the smallest ℓ for which δ^* exceeds zero, we know we have found an optimal min-edge factory.

We can find the unknown length ℓ^* of a min-edge factory in two phases, as follows. The first phase uses doubling, starting from $\ell = 1$, multiplying ℓ by a factor of 2 in the next iteration, where each iteration solves an MILP instance of the above form, until δ^* for the current limit ℓ is greater than zero. This yields an upper bound u on ℓ^*, satisfying $u < 2\ell^*$. The second phase then performs binary search on interval $[0, u]$, again solving MILP instances as above, to find ℓ^*.

To find a min-edge factory, whose optimal number of reactions is k, this solves just $\Theta(\log k)$ instances of the above MILP. For a hypergraph of n vertices and m hyperedges, each MILP instance has $\Theta(m)$ variables and $\Theta(m+n)$ constraints.

Extending to Weighted Factories. We can extend this further to factories with positively *weighted* reactions. We use length constraint $\sum_e \omega(e)\, x_e \leq \ell$, for real ℓ. Then given upper bound u on the shortest factory length, such as $\sum_e \omega(e)$, we perform bisection on $[0, u]$ to find optimal length ℓ^*. This yields a shortest factory, for machine precision ϵ, after solving $O\big(\log(u/\epsilon)\big)$ instances of MILPs.

3 Characterizing Optimal Factories

We next give a complete *characterization* of the structure of shortest factories, that captures for the first time exactly which sets of reactions correspond to optimal factories. We then relate *hyperpaths* to factories, showing they actually comprise a subclass of metabolic factories. Finally we build on this characterization to develop the first practical algorithm for optimal *nondegenerate* factories.

3.1 The Structure of Shortest Factories

To simplify our structural characterization, we first note that the multiple-source and multiple-target factory problem can be reduced to a simpler *single-source* and *single-sink* version, where both problems are under accumulation. (We omit the reduction due to page limits.) Consequently, the s,t-factory problem with a single source s and a single sink t that we work with below is fully general—and leads to a cleaner characterization.

We next define the notions of cuts, crossing a cut, reachability, cycles, and intact sets that are essential to the characterization of factories. (The following terminology is in distinction to the notions of strongly-crossing a cut and strong-reachability that are defined for hyperpaths in Sect. 3.2.)

An s,t-*cut* of hypergraph $G=(V,E)$ with source $s\in V$ and sink $t\in V-\{s\}$ is a bipartition $(C,\overline{C})$ of its vertices V, where $C\subseteq V$ and $\overline{C}:=V-C$, with $s\in C$ and $t\in\overline{C}$. We often refer to such a cut by just specifying its source-side C.

Hyperedge e *weakly-crosses* s,t-cut C if $\text{tail}(e)\not\subseteq\overline{C}$ and $\text{head}(e)\not\subseteq C$. In other words, hyperedge e weakly-crosses source-sink cut C if some vertex in $\text{tail}(e)$ is on the source-side C, while some vertex in $\text{head}(e)$ is on the sink-side $\overline{C}$. A set of hyperedges F *weakly-crosses* C if some $e\in F$ weakly-crosses C.

Vertex w is *weakly-reachable* from vertex v by hyperedges $F\subseteq E$ if $v=w$, or recursively, if $w\in\text{head}(e)$ for a hyperedge $e\in F$ with a vertex in $\text{tail}(e)$ that is weakly-reachable from v by F. We also say F *weakly-reaches* w from v. Similarly v is *weakly-backward-reachable* from w if F weakly-reaches w from v.

A *cycle* is a minimal set of hyperedges $F\subseteq E$ where for all distinct $e,f\in F$ both of the following hold: F weakly-reaches some vertex in $\text{tail}(f)$ from some vertex in $\text{head}(e)$, and vice versa F weakly-reaches some vertex in $\text{tail}(e)$ from some vertex in $\text{head}(f)$. In other words, in a cycle, for every pair of hyperedges, both are weakly-reachable from the other within the cycle (while this does not hold for any proper subset of the cycle).

A set $F\subseteq E$ is s,t-*intact* with respect to source s and sink t if F has an in-edge to t, and an in-edge to every vertex other than s that is touched by F. In other words, an intact set is closed with respect to in-edges, as the sink and every vertex it touches other than the source has an in-edge within the set.

We now define the three solution classes that can arise as shortest factories.

Definition 2 (Trails, Whirls, and Eddies). *Consider a directed hypergraph $G=(V,E)$ with source $s\in V$, sink $t\in V-\{s\}$, and hyperedge subset $F\subseteq E$.*

An s,t-trail *is an s,t-intact set F that weakly-crosses every s,t-cut.*

An s,t-whirl *is an s,t-intact set F that does* not *weakly-cross every s,t-cut, but contains a cycle $C\subseteq F$ that touches sink t and does not touch source s.*

An s,t-eddy *is an s,t-intact set F that does* not *weakly-cross every s,t-cut, does* not *contain a cycle touching t and not s, but contains a cycle $C\subseteq F$ that touches neither s nor t yet touches a vertex from which F weakly-reaches t.* □

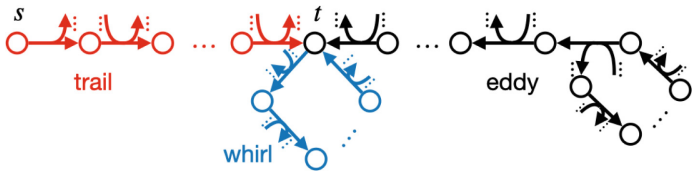

Fig. 1. The three classes of factories. Every shortest factory is either a *trail* (shown in red), *whirl* (in blue), or *eddy* (in black). Conversely, every trail, whirl, or eddy is a shortest factory for some edge weights and stoichiometries. A general factory instance can always be reduced to an equivalent one with a single source s and a single target t. (Color figure online)

Figure 1 illustrates these three classes. (Hyperedges are drawn to indicate multiple head- and tail-vertices.) These classes are distinct, and together they capture all s,t-intact sets (as the proof below shows).

We can now state our main theorem, which completely characterizes the structure of the active hyperedges of shortest factories for *general* edge weights and stoichiometries, where edge weights are arbitrary reals that can be negative.

(Due to page limits, we only give sketches of the proofs for the theorems in this proceedings paper. Full proofs will appear in the journal paper.)

Theorem 1 (Characterization of Shortest Factories). *Let $G = (V, E)$ be a hypergraph with source $s \in V$ and sink $t \in V - \{s\}$. Then $F \subseteq E$ is the set of active hyperedges of a shortest s,t-factory in G under accumulation for some edge weights and stoichiometries if and only if F is an s,t-trail, -whirl, or -eddy.*

Proof. (*Sketch*) For the *forward* implication, accumulation forces the the active hyperedges F of an s,t-factory to be s,t-intact. Start at sink t and collect the set $R \subseteq V$ of vertices weakly-backward-reachable from t by F. If $s \in R$, we can show F weakly-crosses every s,t-cut, so F is an s,t-trail. Otherwise when $s \notin R$, set F does not weakly-cross all s,t-cuts, and on collecting R depth-first backward from t, the backward search must encounter a cycle $C \subseteq F$; this implies F is an s,t-whirl or -eddy, which proves the forward implication.

For the *reverse* implication, given any s,t-intact set F, we construct flux f, edge weights ω, and stoichiometries M, as follows. For the flux, $f_e = 1$ if $e \in F$; otherwise $f_e = 0$. For the weights, $\omega(e) = -1$ if $e \in F$; otherwise $\omega(e) = 1$. For the stoichiometries, $M = (r_{ve})$ for all $v \in V$ and $e \in E$, with $r_{ve} := r^+_{ve} - r^-_{ve}$, where: $r^+_{ve} = 1/\big|\text{in}(v) \cap F\big|$ if $e \in \text{in}(v) \cap F$, otherwise $r^+_{ve} = 0$; and $r^-_{ve} = 1/\big|\text{out}(v) \cap F\big|$ if $e \in \text{out}(v) \cap F$, otherwise $r^-_{ve} = 0$. For this flux f, its active hyperedges are exactly set F, and the flux into sink t is nonzero (as F is s,t-intact). One can show that this flux f satisfies accumulation for these stoichiometries M, and that for these edge weights ω, its active hyperedges F are a minimum-weight s,t-factory. Thus F is the active hyperedge set of a shortest s,t-factory f under accumulation for edge weights ω and stoichiometries M. Finally any s,t-trail, -whirl, or -eddy is an s,t-intact set, which proves the reverse implication. □

Among the three classes of solutions in Theorem 1, only a *trail* corresponds to a legitimate factory that produces the targets by supplying the sources.

A *whirl*, for physically-valid stoichiometries that conserve mass, corresponds to an equilibrium solution that just maintains the existing amount of targets without increasing their production on supplying sources.

An *eddy*, whose active edges produce a nonzero amount of the targets without consuming sources, is impossible under accumulation unless its reactions fail to conserve mass, which can only arise when the network has been misannotated with erroneous physically-invalid stoichiometries.

In brief, Theorem 1 reveals that a shortest factory solver may return a legitimate solution in the form of a trail—or a *degenerate solution* in the form of a whirl or eddy, both of which have been overlooked by all prior approaches to shortest factories, and potentially output as purported solutions. Later Sect. 3.3 presents the first algorithm that finds an *optimal trail*—guaranteeing it delivers an optimal nondegenerate factory.

3.2 Hyperpaths Are Factories

An unexpected consequence of the characterization in Theorem 1 is that every *hyperpath* is a factory—for *some* choice of stoichiometries. Here we show below in Theorem 2 a much stronger result: that given a metabolic network G with stoichiometries M, every s,t-hyperpath is an s,t-factory under accumulation—for the *fixed* stoichiometries M of G. In other words, surprisingly, hyperpaths are actually a *subclass* of factories.

Formally, an s,t-*hyperpath* is a minimal set $P = \{e_1, \ldots, e_k\}$ of hyperedges that can be ordered so that $\text{tail}(e_1) = \{s\}$, $\text{head}(e_k) \supseteq \{t\}$, and for all $1 < i \leq k$, hyperedge e_i satisfies $\text{tail}(e_i) \subseteq \{s\} \cup \bigcup_{1 \leq j < i} \text{head}(e_j)$. We say P *strongly-reaches* t from s. Hyperedge e *strongly-crosses* cut C if $\text{tail}(e) \subseteq C$ and $\text{head}(e) \not\subseteq C$. Set $S \subseteq E$ strongly-crosses cut C if some hyperedge $e \in S$ does. Equivalently, s,t-hyperpaths are exactly minimal sets that strongly-cross all s,t-cuts [16,17].

Theorem 2 (Hyperpaths Are Factories). *Consider a directed hypergraph $G = (V, E)$ with source $s \in V$, sink $t \in V - \{s\}$, stoichiometries M, and any s,t-hyperpath $P \subseteq E$. Then P is the set of active hyperedges of an s,t-factory in G for M under accumulation.*

Proof. (*Sketch*) Given s,t-hyperpath P, we construct a flux f whose active hyperedges are exactly P with nonzero flux into t that satisfies accumulation under $M = (r_{ve})$. The construction of f processes the edges of $P = \{p_1, \ldots, p_k\}$, making use of their ordering $p_1 \prec p_2 \prec \ldots \prec p_k$ given by the definition of a hyperpath, where it processes them in *reverse order* as successively $p_k, \ldots, p_2, p_1$. For each hyperedge $e \in E$, define the following (possibly empty) subset of its head by $H(e) := \big\{v \in \text{head}(e) : r_{ve} > 0\big\} - \{s,t\}$. Let ϵ be any real value with $\epsilon > 0$.

Before processing the hyperedges of P, initialize the flux to $f_e = 0$ for all $e \in E$. Then when processing the next hyperedge $e \in P$ in reverse order, set its flux by,

$$f_e \;:=\; \max\left\{\epsilon,\; \max_{v \,\in\, H(e)} \left\{ -\sum_{h \in P \,:\, h \succ e} r_{vh}\, f_h \Big/ r_{ve} \right\}\right\}. \tag{1}$$

On setting f_e by the above rule, we can show accumulation holds for all $v \in H(e)$.

With each hyperedge $e = p_i \in P$, associate the following set $W_i \subseteq V$, consisting of all vertices that are only in H-sets of the hyperedges $p_i, p_{i+1}, \ldots, p_k$ processed up through e: namely, $W_i := \bigcup_{h \in P\,:\, h \succeq e} H(h) - \bigcup_{g \in P\,:\, g \prec e} H(g)$.

We can then prove by induction for decreasing $i \in \{k, \ldots, 2, 1\}$ that after processing $e = p_i \in P$ by setting f_e using (1), accumulation is satisfied for all $v \in W_i$.

Consequently after processing all $e \in P$, accumulation is satisfied for every vertex in W_1, which is all vertices touched by P except s and t. Any vertex not touched by P trivially satisfies accumulation, as it has zero flux on all incident hyperedges. When processing is finished, $f_e \geq \epsilon > 0$ for $e \in P$, so there is nonzero flux into t; also $f_e = 0$ for $e \notin P$. Thus f is an s,t-factory under accumulation for G whose active hyperedges are exactly hyperpath P. □

An immediate consequence of Theorem 2 is that for any edge weights ω, the length of a shortest s,t-factory is always at most the length of a shortest s,t-hyperpath. So a min-edge factory can potentially use *fewer reactions* than a hyperpath (but never more). Furthermore factories can potentially exist for *more sets* of sources and targets than hyperpaths (but never fewer). The latter is demonstrated empirically in Sect. 4.1 through Table 2.

3.3 Guaranteeing Nondegeneracy

We now give the first algorithm that is guaranteed to find an optimal *nondegenerate* factory under accumulation, which by our characterization theorem is an optimal *trail*. This relies on a parameter-free algorithm for correctly finding a shortest factory such as from Sect. 2.2 (that may return a degenerate solution), which the nondegenerate algorithm calls as a subroutine.

Our approach to finding a shortest s,t-trail in essence generates *next-best* s,t-factories $f^{(1)}, f^{(2)}, \ldots$ in order of increasing length $\omega(f^{(1)}) \leq \omega(f^{(2)}) \leq \cdots$, stopping at the first $f^{(i)}$ whose active hyperedges form an s,t-trail. This $f^* = f^{(i)}$ is a *shortest* s,t-trail, hence f^* is output as an optimal nondegenerate s,t-factory.

Testing whether the active hyperedges F of a factory under accumulation form an s,t-trail is equivalent to checking whether F weakly-reaches t from s. (Under accumulation, the active hyperedges F of an s,t-factory are s,t-intact. So it suffices to check whether F weakly-crosses all s,t-cuts, which is equivalent to weak-reachability.) Determining whether t is weakly-reachable from s by F can be done in time linear in the total size of hyperedge set F (see [10,13]). So the key is how to generate factories by increasing length to find a shortest trail.

Given a shortest factory for a problem-instance $\mathcal{P}$ whose active hyperedges F are a degenerate *whirl* or *eddy*, the next-best factory for $\mathcal{P}$ whose active hyperedges $F' \neq F$ are a *trail* must either: (1) use a proper *subset* of F, and other hyperedges outside F to weakly-reach t from s; or (2) use a proper *superset* of F. Hence we can find the best trail for $\mathcal{P}$ by solving these two subproblems (1) and (2), which we call respectively $\mathcal{P}_F^{\subseteq}$ and $\mathcal{P}_F^{\supseteq}$, finding a shortest trail for both of $\mathcal{P}_F^{\subseteq}$ and $\mathcal{P}_F^{\supseteq}$, and returning the better of these two trails.

Table 1. Dataset Summaries

	C_Rud		S_Mue		B_Aph		B_Cic		S_Cer		H_Sap		E_Col		Reactome	
Vertices	263		314		460		700		936		1,618		1,877		20,458	
Hyperedges	229		273		447		755		1,250		2,132		2,999		11,802	
Sources	40		45		45		58		128		171		65		8,296	
Targets	44		48		51		67		227		344		73		5,066	
	med	max	med	max	med	max	med	max	med	max	med	max	med	max	med	max
Tail size	2	4	2	4	2	6	2	6	1	2	1	2	2	7	2	26
Head size	2	5	2	5	2	6	2	6	1	3	1	3	2	95	1	28
In-degree	1	41	1	49	1	67	1	156	1	15	1	13	1	806	1	1,056
Out-degree	1	64	1	72	1	104	1	142	1	8	1	18	1	511	1	1,167

We can solve subset subproblem $\mathcal{P}' := \mathcal{P}_F^{\subsetneq}$, and superset subproblem $\mathcal{P}' := \mathcal{P}_F^{\supsetneq}$, by constructing a modified MILP for the parameter-free algorithm, and examining its solution F'. (We omit details of the modified MILP due to page limits.) If F' is a trail, we can show it is an optimal solution to subproblem $\mathcal{P}'$. Otherwise, we recursively solve the two further subproblems $(\mathcal{P}')_{F'}^{\subsetneq}$ and $(\mathcal{P}')_{F'}^{\supsetneq}$.

We implement this whole process using a heap H of such subproblems, where the priority of subproblem $\mathcal{P}$ on H is the *length* (or total weight) of the best (possibly degenerate) factory for $\mathcal{P}$ found by the parameter-free algorithm. The nondegenerate algorithm repeatedly extracts from H the subproblem $\mathcal{P}$ of minimum priority, and fetches the corresponding optimal factory f^* for $\mathcal{P}$. If f^* is a trail, the nondegenerate algorithm halts and outputs f^*. Otherwise, subproblems $\mathcal{P}_F^{\subsetneq}$ and $\mathcal{P}_F^{\supsetneq}$ are inserted into H, and the algorithm continues.

This heap-based approach can find an optimal nondegenerate factory quickly, when shortest factories have few hyperedges, and there are few near-optimal factories—both of which are typically the case.

4 Experimental Results

We now present results from computational experiments on biological benchmark datasets. We highlight an example of a shortest factory instance where the prior state-of-the-art fails, and evaluate the speed of our parameter-free algorithm. (In addition, we also performed experiments that investigated imputing unit stoichiometry ratios, and further differences between the parameter-based and parameter-free approaches, which will appear in the full journal paper.)

4.1 Experimental Setup

Datasets. We evaluate our parameter-free algorithm for shortest factories on eight standard datasets, which we now briefly describe. (For a full description of these datasets, and their transformation into hypergraphs, see [12].) Seven of the datasets are metabolic networks for model organisms from MetExplore [3]. We use an abbreviation of the organism name to identify these datasets (namely, first letter of genus, underscore, followed by first three letters of species). The final dataset contains curated human signaling pathways from Reactome [7].

Table 2. Target Instance Feasibility

	C_Rud	S_Mue	B_Aph	B_Cic	vS_Cer	H_Sap	E_Col	Reactome
Target instances	40	48	51	67	142	344	73	5,066
Factory, accumulation	6	17	25	40	131	273	48	3,955
Factory, conservation	0	0	0	13	127	235	1	1,632
Hyperpath	0	0	2	1	129	267	1	2,432

We consider any vertex with no in-edges a *source*, and any vertex with no out-edges a *target*. A problem *instance* then involves finding a factory (or hyperpath) from all of the sources to a given target. (When computing hyperpaths, we created a supersource, and a zero-weight hyperedge with the supersource as its tail, and all source vertices in its head.)

Table 1 gives statistics on the hypergraphs constructed for each dataset, listed in order of increasing size. Overall, the hypergraphs tend to have fewer hyperedges than vertices, suggesting potentially low connectivity between nodes.

Table 2 reports for each dataset the number of instances having a factory (either under accumulation or conservation), or a hyperpath. As expected, there are more instances with factories under accumulation than under conservation, and more instances with factories under accumulation than hyperpaths.

Implementation. Our new tool `Freeia` [18] implements the parameter-free algorithm for min-edge factories from Sect. 2.2, though it first runs two heuristics described below to quickly: (1) check feasibility, or the existence of any factory; and (2) bound the minimum number of active hyperedges, potentially tightly.

The first heuristic tests feasibility by solving the parameter-free MILP from Sect. 2.2, but without a length constraint on the number of active hyperedges. If the objective function value of an optimal solution with no length constraint is zero, `Freeia` reports no factory exists that produces the targets.

The second heuristic has two steps. It first runs `Odinn` [11] (the current state-of-the-art for optimal factories, which may exclude valid factories due to its parameter choice), to get an upper bound $\widetilde{k}$ on the number of active hyperedges in a min-edge factory. Then it solves the parameter-free MILP from Sect. 2.2 where its length constraint has $\ell := \widetilde{k} - 1$. If the objective function under this constraint is zero, we know `Odinn`'s solution is optimal, and is output by `Freeia`. Otherwise, `Freeia` runs the full parameter-free algorithm with binary search.

`Freeia` comprises around 300 lines of Python code. For the experiments, we used `Mmunin` [16] to compute shortest hyperpaths. For directed hypergraph representations, we used Halp (https://github.com/Murali-group/halp). MILPs were solved using `CPLEX` 12.6, run on an M1 processor with 8 GB of memory.

4.2 `Freeia` Finds Factories Missed by the Prior State-of-the-Art

The current state-of-the-art for min-edge factories (that produce the targets using the fewest reactions) is `Odinn` [11,12], which has a target production

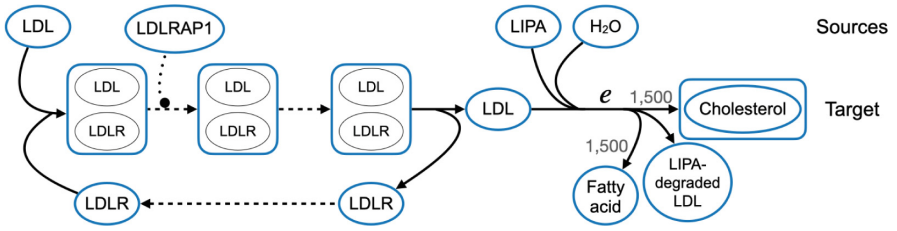

Fig. 2. Shortest factory to the target "Cholesterol" in `Reactome`. Sources are shown along the top of the figure; the target is at the far right. Dashed hyperedges represent transport between cellular compartments. The dotted curve from LDLRAP1 ending in a disc indicates it is a positive regulator. All stoichiometry ratios are unit, except for hyperedge e where "Fatty acid" and "Cholesterol" both have stoichiometry ratio 1,500.

parameter ϵ (settable by the user) that specifies the minimum amount of all targets that must be produced by a legitimate factory. Assigning parameter ϵ can be difficult, since setting it too high excludes factories that produce the target in a nonzero amount less than ϵ, while setting it too low allows for numerical tolerances within `CPLEX` to exclude valid factories. Other methods for finding optimal factories also have an equivalent parameter that instead upper bounds the maximum allowable flux on any hyperedge. `Freeia` avoids these issues, and includes all valid factories in its space of feasible solutions.

Figure 2 highlights an instance where `Freeia` finds an optimal factory, yet `Odinn` (being parameter-based) reports no valid factory exists. The figure shows a shortest factory to "Cholesterol" in the pathway "Transport of small molecules" from `Reactome`. Sources are along the top of the figure, and the target is at the far right. Dashed hyperedges represent transport from one cellular compartment to another. The dotted portion of the hyperedge involving LDLRAP1 indicates it acts as a positive regulator of this reaction. All stoichiometries are unit, except on the final hyperedge to the target, where "Fatty acid" and "Cholesterol" each have stoichiometry 1,500. Note that this factory is not a hyperpath, due to the cycle involving LDLR (and no hyperpath for this target exists).

Biologically, the reactions in this pathway lead to the accumulation of excess cholesterol in atherosclerosis [5]. The process begins with the migration of low density lipoproteins (LDLs) across an injured artery endothelium, after binding to the LDL receptor (LDLR). Once transported into the subendothelial space, these LDLs undergo hydrolysis by lysosomal acid lipase (LIPA), causing a massive release of cholesterol and fatty acids.

`Odinn` fails to return a valid factory for this instance due to numerical issues in `CPLEX` caused by the high stoichiometry ratio for Cholesterol on its in-edge e. (We note that this stoichiometry ratio may be an incorrect annotation.) The default value for the target production parameter in `Odinn` is $\epsilon := 5 \times 10^{-4}$, meaning a factory is feasible only if it produces target t in quantity at least ϵ. This high stoichiometry ratio allows sufficient target production with a very small amount of flux on final hyperedge e, namely $f_e = \epsilon / r_{te} = 3 \times 10^{-7}$. In `Odinn`,

Table 3. Running Time

	C_Rud		S_Mue		B_Aph		B_Cic		S_Cer		H_Sap		E_Col		Reactome	
	med	max	med	max	med	max	med	max	med	max	med	max	med	vmax	med	max
Freeia, accum.	0.1	0.1	0.1	0.2	0.1	0.5	0.2	28	0.1	0.4	0.3	2.3	37	3,168	4.4	4,141
Odinn, accum.	0.1	0.1	0.1	0.1	0.1	0.2	0.1	2.3	0.1	0.1	0.1	0.2	3.1	2,943	3.1	6.6
Freeia, conserv.	†	†	†	†	†	†	0.2	0.5	0.2	47	0.4	75	0.7	0.7	4.3	97
Odinn, conserv.	†	†	†	†	†	†	0.1	0.1	0.1	0.3	0.1	0.5	0.2	0.2	2.9	4.1
Mmunin	†	†	†	†	0.1	0.1	0.1	0.1	0.1	525	0.1	314	0.1	0.1	9	776

† Either no factory under conservation, or no hyperpath, exists for any target in this dataset.
Time is in seconds.

each hyperedge e in the hypergraph has both a real-valued variable f_e, giving the flux on e, and an integer-valued variable x_e, indicating whether e is active. (Odinn minimizes its number of active hyperedges.) These variables are related by constraint $x_e \geq f_e$, which should force $x_e = 1$ when $f_e > 0$. Unfortunately due to numerical tolerances in CPLEX, small values for f_e do not force $x_e = 1$, so hyperedge e is not considered active, and Odinn fails to output a valid factory.

Strikingly, we experimentally confirmed that there is *no default value* for the target production parameter ϵ at which Odinn returns an optimal factory for all instances in Reactome. The largest possible value for ϵ was calculated using Freeia, where the objective function value of our parameter-free MILP gives an upper bound on target production for any factory with an optimal number of reactions. (For any higher ϵ-value, Odinn either returns a suboptimal factory that creates the target in a larger quantity, or reports no factory exists.) The smallest value for ϵ was found by running Odinn and observing when CPLEX returned an invalid factory due to numerical errors. The following table lists, for various ϵ-values, the number of Reactome instances on which Odinn experiences the following *failure* modes: finding an invalid factory, when ϵ is too low; or a suboptimal factory, or no factory, when ϵ is too high.

Target production parameter ϵ	$\leq 10^{-5}$	5×10^{-4}	0.0125	0.025	0.05	0.1	0.2	0.4	0.8
Odinn failures (out of 5,066 instances)	all	1	2	15	35	143	378	714	1,040

More precisely, Odinn fails on at least one Reactome instance for *every possible ϵ-value*: it fails on the instance from Fig. 2 for $\epsilon \leq 0.03$, and fails on a separate instance for $\epsilon \geq 0.0125$. In distinction, Freeia never returns invalid or suboptimal solutions, or reports no solution exists on an instance with a valid factory.

4.3 Speed of Computing Parameter-Free Factories

Computing robust optimal factories with Freeia is fast in practice, with a median running time across all instances of under five seconds. Table 3 compares running times on datasets for: Freeia, our new parameter-free tool; Odinn, the prior state-of-the-art for optimal factories; and Mmunin, the current state-of-the-art for shortest hyperpaths. (Running times are reported in seconds.) For

all datasets except `E_Col` and `Reactome`, the `Freeia` tool has median running time under a second, with its maximum time just over a minute. For the more challenging datasets, `Freeia` maintains a fast median time, but its maximum time rises to just over an hour for an isolated instance in `Reactome`. The time for these longer-running instances is typically dominated by solving the MILP from the second heuristic implemented in `Freeia`, which checks if a valid factory exists with one-fewer active hyperedge than `Odinn`'s factory. We noticed that the computational cost of this MILP typically scales with the number of active hyperedges in an optimal factory, and that the shortest factory for any instance taking more than 1,000 seconds contains at least 20 hyperedges.

Surprisingly, `Freeia` tends to be faster than finding shortest hyperpaths with `Mmunin`. This is interesting, since by Theorem 2 every hyperpath is a factory, which implies that for any instance its space of feasible factories is larger than of feasible hyperpaths. This difference in running time appears due to algorithmic differences in `Mmunin` and `Freeia`, as `Mmunin` typically solves more MILPs per instance than `Freeia` (though `Freeia`'s MILPs occasionally take longer to solve).

5 Conclusion

We have presented the first *robust algorithm* for optimal factories, that is free of parameter settings and guarantees nondegeneracy. We also for the first time characterized the graph-theoretic structure of shortest factories, establishing hyperpaths are a subclass of factories. Comprehensive experiments demonstrate our algorithm is fast in practice, and finds solutions missed by the prior state-of-the-art.

Further Research. A major open problem is the characterization of shortest factories for the case of *positive edge weights* (which arise under both parsimony and maximum likelihood). While there is likely no constant-factor approximation algorithm for Shortest Factory (since its NP-completeness proof [12] shows it is as hard to approximate as Set Cover), a *fast heuristic* for shortest factories (like for shortest hyperpaths [13]) would be useful, since all current approaches solve mixed-integer linear programs, whose time could potentially explode in practice.

Acknowledgments. We thank Anna Ritz for sharing the `BioPax` parser, and the anonymous referees for their useful comments. Research supported by the National Science Foundation through grants CCF-1617192 and IIS-2041613 to JK.

References

1. Acuña, V., Milreu, P.V., Cottret, L., et al.: Algorithms and complexity of enumerating minimal precursor sets in genome-wide metabolic networks. Bioinformatics **28**(19), 2474–2483 (2012)
2. Andrade, R., Wannagat, M., Klein, C.C., et al.: Enumeration of minimal stoichiometric precursor sets in metabolic networks. Alg. for Mol. Bio. **11**(1), 25 (2016). https://doi.org/10.1186/s13015-016-0087-3

3. Cottret, L., Frainay, C., Chazalviel, M., et al.: MetExplore: collaborative edition and exploration of metabolic networks. Nucleic Acids Res. **46**(W1), W495–W502 (2018)
4. Cottret, L., Vieira Milreu, P., Acuña, V., et al.: Enumerating precursor sets of target metabolites in a metabolic network. In: Proceedings of the 8th Workshop on Algorithms in Bioinformatics (WABI), pp. 233–244 (2008)
5. Dubland, J.A., Francis, G.A.: Lysosomal acid lipase: at the crossroads of normal and atherogenic cholesterol metabolism. Front. Cell Dev. Bio. **3**, 3 (2015)
6. Italiano, G.F., Nanni, U.: Online maintenance of minimal directed hypergraphs. Department of Computer Science, Columbia University, Tech. rep. (1989)
7. Joshi-Tope, G., Gillespie, M., Vastrik, I., et al.: Reactome: a knowledgebase of biological pathways. Nucleic Acids Res. **33**, D428-432 (2005)
8. Klamt, S., Haus, U.U., Theis, F.: Hypergraphs and cellular networks. PLoS Comput. Biol. **5**(5), e1000385 (2009)
9. Krieger, S.: Algorithmic Inference of Cellular Reaction Pathways and Protein Secondary Structure. PhD dissertation, Department of Computer Science, The University of Arizona (July 2022)
10. Krieger, S., Kececioglu, J.: Fast approximate shortest hyperpaths for inferring pathways in cell signaling hypergraphs. In: Proceedings of the 21st ISCB Workshop on Algorithms in Bioinformatics (WABI). Leibniz International Proceedings in Informatics, vol. 201, pp. 1–20 (2021)
11. Krieger, S., Kececioglu, J.: `Odinn`: optimal minimum-hyperedge factories in metabolic networks with negative regulation, version 1.0 (2021). http://odinn.cs.arizona.edu
12. Krieger, S., Kececioglu, J.: Computing optimal factories in metabolic networks with negative regulation. Bioinformatics **38**(Suppl. 1), i369–i377 (2022). Proceedings of the 30th ISCB Conference on Intelligent Systems for Molecular Biology (ISMB)
13. Krieger, S., Kececioglu, J.: Heuristic shortest hyperpaths in cell signaling hypergraphs. Algorithms Mol. Biol. **17**(1), 12 (2022). https://doi.org/10.1186/s13015-022-00217-9
14. Krieger, S., Kececioglu, J.: `Hhugin`: hypergraph heuristic for general shortest source-sink hyperpaths, version 1.0 (2022). http://hhugin.cs.arizona.edu
15. Krieger, S., Kececioglu, J.: `Mmunin`: integer-linear-programming-based cutting-plane algorithm for shortest source-sink hyperpaths, version 1.0 (2022). http://mmunin.cs.arizona.edu
16. Krieger, S., Kececioglu, J.: Computing shortest hyperpaths for pathway inference in cellular reaction networks. In: Proceedings of the 27th Conference on Research in Computational Molecular Biology (RECOMB), LNBI 13976, pp. 155–173. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-29119-7_10
17. Krieger, S., Kececioglu, J.: Shortest hyperpaths in directed hypergraphs for reaction pathway inference. J. Comput. Biol. **30**(11), 1–28 (2023)
18. Krieger, S., Kececioglu, J.: `Freeia`: robust optimal factories in metabolic reaction networks, version 1.0 (2023). http://freeia.cs.arizona.edu
19. Li, Y., McGrail, D.J., Latysheva, N., et al.: Pathway perturbations in signaling networks: linking genotype to phenotype. Sem. in Cell Dev. Bio. **99**, 3–11 (2020)
20. Ritz, A., Avent, B., Murali, T.: Pathway analysis with signaling hypergraphs. IEEE/ACM Trans. Comp. Bio. Bioinf. **14**(5), 1042–1055 (2017)
21. Ritz, A., Murali, T.: Pathway analysis with signaling hypergraphs. In: Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB), pp. 249–258 (2014)

22. Ritz, A., Tegge, A.N., Kim, H., et al.: Signaling hypergraphs. Trends Biotechnol. **32**(7), 356–362 (2014)
23. Sharan, R., Ideker, T.: Modeling cellular machinery through biological network comparison. Nat. Biotechnol. **24**(4), 427–433 (2006)
24. Vidal, M., Cusick, M.E., Barabási, A.L.: Interactome networks and human disease. Cell **144**(6), 986–998 (2011)
25. Zarecki, R., Oberhardt, M.A., Reshef, L., et al.: A novel nutritional predictor links microbial fastidiousness with lowered ubiquity, growth rate, and cooperativeness. PLoS Comput. Biol. **10**(7), 1–12 (2014)