nature machine intelligence

Article

https://doi.org/10.1038/s42256-025-01069-9

Unifying multi-sample network inference from prior knowledge and omics data with CORNETO

Received: 9 December 2024

Accepted: 2 June 2025

Published online: 22 July 2025

Check for updates

Pablo Rodriguez-Mier[1], Martin Garrido-Rodriguez[1,2,4], Attila Gabor[1,4] & Julio Saez-Rodriguez[1,3]

Understanding biological systems requires methods that extract interpretable insights from omics data. Networks offer a natural abstraction by representing molecules as vertices and their interactions as edges, providing a foundation for constructing context-specific models tailored to particular conditions—an essential step in many biological analyses. Most existing approaches fall into one of two categories: machine learning methods, which offer strong predictive power but lack interpretability and require large datasets, and knowledge-based methods, which are more interpretable but designed for analysing individual samples and difficult to generalize. Here we present CORNETO, a unified mathematical framework that generalizes a wide variety of methods that learn biological networks from omics data and prior knowledge. CORNETO reformulates these methods as mixed-integer optimization problems using network flows and structured sparsity, enabling joint inference across multiple samples. This improves the discovery of both shared and sample-specific molecular mechanisms while yielding sparser, more interpretable solutions. CORNETO supports a range of prior knowledge structures, including undirected, directed and signed (hyper)graphs. It extends a broad class of approaches, ranging from Steiner trees to flux balance analysis, within a unified optimization-based interface. We demonstrate CORNETO's utility across diverse biological contexts, including signalling, metabolism and integration with biologically informed deep learning. We provide CORNETO as an open-source Python library for flexible network modelling.

Understanding biological function arising from molecular interactions is central to systems biology[1]. These systems are often represented as networks, where nodes denote entities (for example, genes or proteins) and edges denote interactions (for example, binding or activation)[2,3]. However, interactions are context dependent, varying across cell types and conditions. This motivates the construction of context-specific biological networks, reflecting functional interactions in specific scenarios through network inference approaches[4,5].

One approach to network inference relies on purely data-driven statistical and machine learning methods[6]. While powerful, these typically require large datasets and multiple data modalities, limiting their application mainly to areas like gene regulation, where extensive single-cell transcriptomics data are available[7,8]. Conversely, for many other data modalities—including bulk transcriptomics, (phospho) proteomics and metabolomics, such large-scale datasets are generally scarce, restricting inference to methods such as correlation networks[9].

[1]Institute for Computational Biomedicine, Heidelberg University, Faculty of Medicine, and Heidelberg University Hospital, Heidelberg, Germany. [2]European Molecular Biology Laboratory, Molecular Systems Biology Unit, Heidelberg, Germany. [3]European Molecular Biology Laboratory, European Bioinformatics Institute, Hinxton, UK. [4]These authors contributed equally: Martin Garrido-Rodriguez, Attila Gabor. e-mail: saezlab@ebi.ac.uk

While straightforward to compute, correlation-based methods capture only statistical associations and offer no causal interpretability. For example, two proteins may be co-expressed across multiple conditions, but this does not establish whether one regulates the other directly or whether both are controlled by a shared upstream regulator[10,11].

Integrating experimental data with biological prior knowledge is an effective alternative, particularly when samples are limited[12]. Prior knowledge networks (PKNs)—structured repositories of known interactions—help to address data scarcity by providing a useful inductive bias, guiding methods towards biologically plausible hypotheses. Network inference then involves identifying a context-specific subnetwork within a PKN, constraining potential interactions to those with established relevance and thereby improving both accuracy and interpretability. Numerous methods use PKNs to infer protein–protein interaction (PPI)[13], signalling[14] and metabolic networks[15], typically by mapping data onto a PKN and extracting relevant subnetworks.

Despite these advances, the field remains fragmented: inference methods are often tailored to specific data types, rely on ad hoc assumptions or are limited to particular network structures (for example, signalling networks[14] or metabolic networks[16]). This heterogeneity complicates generalization, systematic benchmarking and the identification of unifying principles. Furthermore, although many studies contain multiple samples (for example, replicates, conditions and timepoints), most PKN-based approaches analyse them independently or in a pairwise manner. Methods analysing samples individually have limited ability to distinguish context-dependent interactions shared across conditions from sample-specific variation[17]. Non-identifiability—where multiple networks can equally explain the data—further complicates network inference[18]. More mechanistic approaches, such as Boolean or differential-equation-based models[19–21], can integrate information from multiple samples. However, they typically require detailed prior knowledge of the network structure, which is often unavailable, and also present scalability issues with larger networks, limiting their applicability.

To address these challenges, we introduce CORNETO (constrained optimization for the recovery of networks from omics), a general, extensible framework for knowledge-driven network inference. CORNETO integrates data across multiple samples, supports various PKN structures (for example, graphs and hypergraphs) and models diverse biological problems (for example, signalling and metabolism) using a single constrained mixed-integer optimization formulation[22]. By casting network inference as a joint optimization problem governed by structured sparsity and network flow principles[23,24], CORNETO reconciles the strengths of prior knowledge with the robustness of multi-sample modelling. Unlike traditional methods, CORNETO performs inference jointly across samples, enabling information sharing to improve the inference while identifying sample-specific subnetworks. This unified framework demonstrates that a priori diverse methods can be viewed as special cases of a common optimization problem, thereby enabling systematic extension, integration and benchmarking. CORNETO bridges data-driven methods[25,26] and traditional PKN-based approaches, which typically focus on individual samples, such as flux balance analysis (FBA), sparse FBA (sFBA), iMAT (Integrative Metabolic Analysis Tool) for metabolic networks[27], CARNIVAL (CAusal Reasoning for Network identification using Integer VALue programming) for signalling[28] or prize-collecting Steiner trees (PCSTs) for protein–protein interactions[29]. We demonstrate how methods for signalling, metabolic and PPI networks can be extended for joint analysis using CORNETO and evaluate performance on both simulated and real datasets. Finally, we show how CORNETO links network inference with machine learning by enabling the construction of optimized, biologically informed neural networks. The open-source Python package implementing this framework is available via GitHub at https://github.com/saezlab/corneto.

Results

Overview

CORNETO provides a unified framework and Python implementation for prior-knowledge-driven network inference from omics data (Fig. 1). It reformulates network inference as a general constrained optimization problem, enabling the joint inference of multiple context specific subnetworks from a PKN. By expressing diverse methods through a common mathematical programming formulation, CORNETO offers flexibility, precise control over biological assumptions, and compatibility with powerful solvers. This approach supports the integration of multiple omics modalities and the inference of a wide range of biologically meaningful substructures such as paths, trees, direct acyclic graphs or subgraphs that best explain the data in light of prior knowledge.

All methods in CORNETO follow a shared structure: importing a PKN, mapping omics data onto the network, and defining method specific constraints and objectives. These are encoded as optimization variables and mathematical expressions that determine which substructures are selected. Because this process is expressed generically in terms of mathematical programming, it is fully decoupled from solver selection and supports rapid prototyping, customization and symbolic manipulation of variables and constraints.

This unified architecture reveals that many widely used network inference methods are special cases of the general optimization framework implemented by CORNETO (Table 1), making it straightforward to extend or improve a wide range of methods using the same underlying optimization principles. Once implemented in CORNETO, any method can be directly extended to multi-sample inference using a structured sparsity-inducing penalty. This allows the joint inference of networks across samples, improving robustness, reducing false positives and capturing both shared and condition-specific features in a principled way.

We illustrate CORNETO's versatility through applications involving method extension or adaptation: (1) extending FBA to multi-sample scenarios; (2) integrating multi-sample omics data with FBA for context-specific metabolic networks; and (3) using multi-sample and multi-omics data for joint inference of signalling networks. The main text focuses on these primary applications, while the Supplementary Information showcases how CORNETO can also be used to implement (4) Steiner tree-based methods for multi-sample protein interaction modules and kinase–substrate interactions in various contexts, and (5) the automatic construction of optimized, biologically informed neural network architectures trained on single-cell transcriptomics. Together, these examples demonstrate CORNETO's support for diverse network-based methods across a range of omics modalities (Table 1).

Unified framework for joint network inference from prior knowledge

CORNETO decomposes network inference into four components: a mapping function ϕ (Fig. 2a), a graph transformation ψ (Fig. 2b), an objective function f and a set of variables and constraints (Fig. 2c) and casts the entire problem as a mixed-integer network-flow formulation. Network flows provide a mathematical framework for assigning quantities to edges under capacity, directionality and conservation constraints. This abstraction lets us capture a wide variety of inference tasks, such as finding shortest paths, extracting minimal subnetworks, balancing metabolic fluxes, inferring signal propagation pathways and more, by defining appropriate costs or scores on edges and then optimizing the flows accordingly.

Starting from omics data $\mathbf{D} \in \mathbb{R}^{m\times n}$ and a hypergraph (PKN) $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the mapping $\phi : (\mathcal{H}, \mathbf{D}) \to \mathcal{H}'$ projects measurements onto $\mathcal{H}$, producing an annotated hypergraph $\mathcal{H}'$. We use $\mathcal{H}$ for general hypergraphs (hyperedges may join multiple vertices) and $\mathcal{G}$ for ordinary graphs (edges join exactly two vertices), which can be viewed as a special case of hypergraphs. This distinction helps to

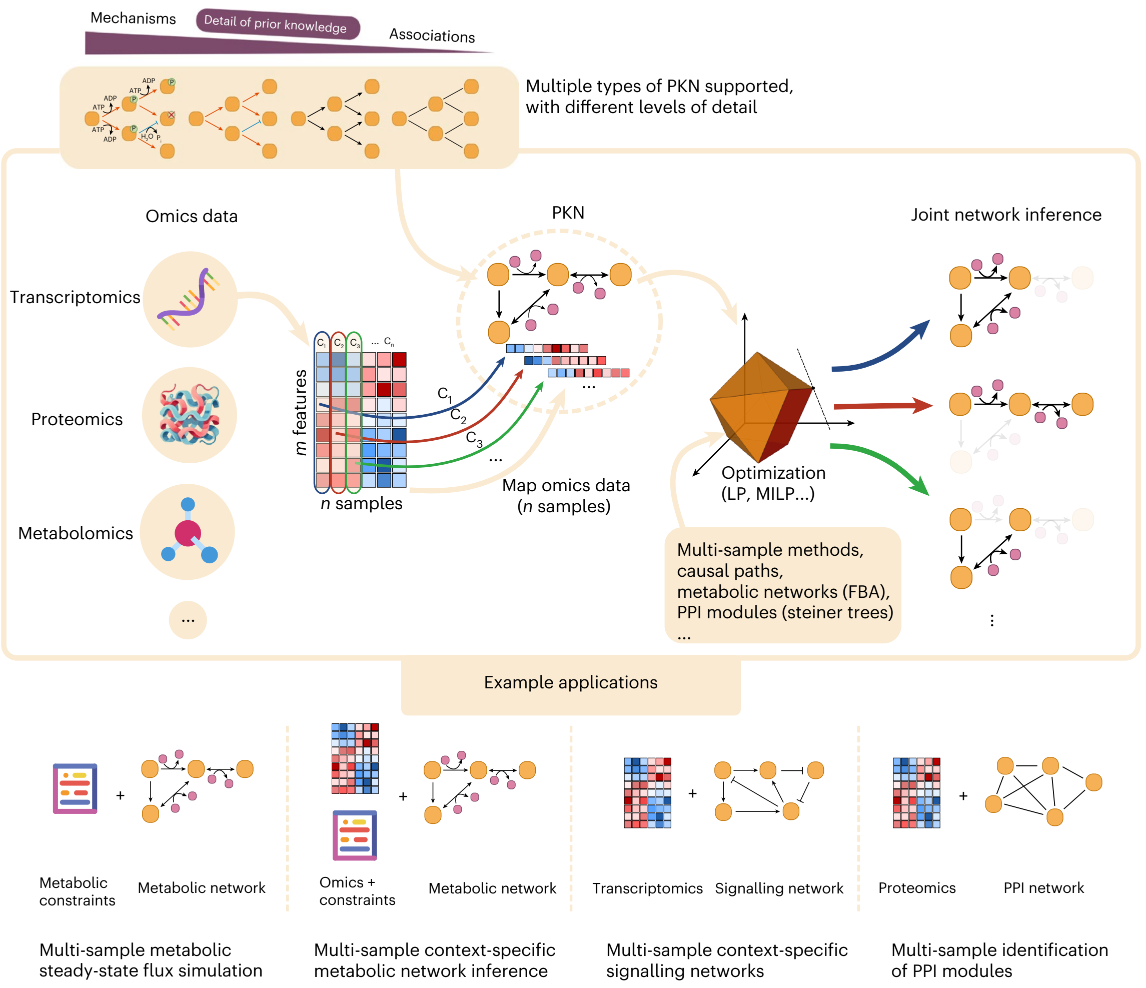
Mechanisms
Detail of prior knowledge
Associations
Multiple types of PKN supported, with different levels of detail
Omics data
Transcriptomics
Proteomics
Metabolomics
...
PKN
m features
n samples
Map omics data (n samples)
Joint network inference
Optimization (LP, MILP...)
Multi-sample methods, causal paths, metabolic networks (FBA), PPI modules (steiner trees) ...
Example applications
Metabolic constraints
Metabolic network
Omics + constraints
Metabolic network
Transcriptomics
Signalling network
Proteomics
PPI network
Multi-sample metabolic steady-state flux simulation
Multi-sample context-specific metabolic network inference
Multi-sample context-specific signalling networks
Multi-sample identification of PPI modules

Fig. 1 | Overview of CORNETO's framework and example applications. CORNETO is a flexible framework that allows users to implement and extend diverse network inference methods by reformulating them as constrained optimization problems. For any method, the process begins by mapping omics data (for example, transcriptomics or proteomics) onto PKNs, which can vary in complexity from simple interaction graphs to detailed metabolic networks. CORNETO then reformulates the method into a specific optimization problem, ensuring that it can be optimally solved using mathematical programming techniques. A key contribution of CORNETO is its ability to enable joint inference across multiple samples, leveraging shared patterns to improve network inference. Using this framework, we have extended several methods and demonstrated their effectiveness in diverse biological use cases. Figure adapted from ref. 14 (Fig. 2) under a Creative Commons license CC-BY 4.0.

indicate when certain methods do not naturally extend to the hypergraph setting, even though they are implemented uniformly within the framework. Transformations are accordingly written as $\mathcal{H}' \to \mathcal{H}''$ or $\mathcal{G}' \to \mathcal{G}''$.

Next, $\psi : \mathcal{H}' \to \mathcal{H}''$ applies preprocessing strategies such as pruning unreachable vertices or irrelevant edges and, most importantly, inserting source and sink edges to enable the injection and extraction of flows in a method-specific manner. This transformation constitutes the first step in reformulating a given method into a flow-based problem. In metabolic networks, these edges may already be present in the PKN (for example, importer and exporter reactions) and, thus, may not be required to guarantee that flows can traverse the network.

Applying ϕ and ψ to each sample yields a collection of transformed hypergraphs, which we merge into a union hypergraph $\mathcal{H}_u$. On $\mathcal{H}_u$, the flow for one sample is a vector $\mathbf{x} \in \mathbb{R}^{|\mathcal{E}_u|}$, whose entries specify the flow of the edges under capacity, directionality and conservation at each vertex; collecting these gives $\mathbf{X} \in \mathbb{R}^{|\mathcal{E}_u| \times n}$. Binary indicators $\mathbf{Y} \in \{0,1\}^{|\mathcal{E}_u| \times n}$ specify which edges carry non-zero flow per sample, and they are used to enforce sparsity across samples.

Flows and indicator variables model the selection of graph structures (paths, trees, direct acyclic graphs or other combinatorial objects), with flows $\mathbf{X}$ and edge indicators $\mathbf{Y}$ determined automatically by optimization, although any known presence or absence of specific edges can be manually enforced in $\mathbf{Y}$. CORNETO builds on these variables and constraints and offers a Python application programming interface (API) for defining linear inequalities $g_i(\mathbf{X}, \mathbf{Y}, \ldots) \leq 0$ and equalities $h_j(\mathbf{X}, \mathbf{Y}, \ldots) = 0$, which together define the feasible set Ω (for example, all valid trees from a PKN). Adding an objective $f : \Omega \to \mathbb{R}$ yields the constrained optimization problem $\mathcal{P} = (f, \Omega)$, whose solution via an exact solver gives the optimal $\mathbf{X}$ and $\mathbf{Y}$ and, thus, the inferred network(s) (Fig. 2d). The generality of the framework allows many methods to be expressed as special cases of constrained optimization. As an illustration, Supplementary Fig. 2 shows how the simple weighted shortest-path algorithm can be formulated using CORNETO's framework. Once implemented in the framework, such methods automatically extend to multi-sample settings without additional effort.

This extension to multi-sample inference is achieved by introducing a structured sparsity regularizer $\lambda \, \|\mathbf{Y}\,\mathbf{1}_n\|_0$, which promotes consistent edge selection of context-specific interactions by jointly penalizing entire edge groups across samples. This results in sparser, more interpretable networks and enhances the detection of shared interactions, because the space of potential interactions that can be selected from the PKN is reduced (Fig. 2e). As with any regularizer, λ is problem specific and should be chosen via hyperparameter tuning.

Table 1 | Overview of the novel methods implemented and evaluated in CORNETO, spanning five different applications for network modelling and inference

			Validation		
Application	**PKN**	**New method**	**Inputs**	**Outputs**	**Strategy**
Genome-scale metabolic modelling and simulation	Hypergraph, (bi)directional hyperedges	Multi-sample sFBA (related: refs. 31,75)	MitoCore v1.01 (ref. 35), metabolic constraints (min. biomass production)	Metabolic fluxes and flux-consistent metabolic subnetworks per sample	Simulated mutants (single enzyme knockouts) as samples, recovery of sparse metabolic networks
	Hypergraph, (bi)directional hyperedges	Multi-sample metabolic flux adjustment (related: refs. 32,39)	MitoCore v1.01 (ref. 35), noisy metabolic fluxes sampled from the PKN	Predicted metabolic fluxes (including missing values) per sample	Cross-validation for predictive accuracy on simulated fluxes under varying noise and percentage of missing values
Context-specific metabolic network inference from omics	Hypergraph, (bi)directional hyperedges	Multi-sample iMAT (related: ref. 27)	Yeast-GEM v8.5.0 (ref. 80), transcriptomics data from single yeast knockout strains[43]	Context-specific metabolic subnetworks per sample	Cross-validation using transcriptomics data (leaving out reaction scores)
Context-specific intracellular signalling inference	Directed graph with signed interactions	Multi-sample CARNIVAL (related: refs. 28,81)	Omnipath[82], patient data from CPTAC (transcriptomics, phosphoproteomics)	Patient-specific intracellular signalling networks	Inference from transcriptomics, validation using phosphoproteomics
Protein–protein and kinase–substrate network inference	Undirected, directed graph	Multi-sample Steiner tree or PCST (related: refs. 83–85)	Omnipath[82], transcriptomics, proteomics, phosphoproteomics	Context-specific tree modules per sample	Simulations (single, multi-sample) to measure error versus tree size, and comparisons with NetworkX
Biologically inspired neural networks	Directed acyclic graph	Min-KPNN (minimal knowledge-primed neural network) (related: refs. 60,86)	Original PKN[60], CROP-seq T cell receptor stimulation[78] from original study	Compressed and trained biologically inspired neural network	Cross-validation classification error on T cell receptor activity from single-cell RNA sequencing data

Application: the type of network inference and modelling task covered in the paper. PKN: the graph structure (for example, hypergraph, directed or undirected) used to model the prior knowledge. New method: the novel multi-sample method implemented in CORNETO. Inputs: inputs used to evaluate the method (typically a PKN + data). Outputs: outputs generated by the method, used to evaluate performance. Strategy: validation strategies used to evaluate the performance of the methods.

We provide more details about the framework in the Methods and Supplementary Information.

In the following sections, we introduce new multi-sample methods in CORNETO and benchmark them against single-sample approaches on simulated and real data.

CORNETO improves FBA modelling

We first use CORNETO to enhance the capabilities of FBA simulation and modelling for multi-sample settings. FBA is widely used to predict metabolic fluxes, assess the impact of genetic or environmental changes on metabolism, and guide metabolic engineering strategies.

We demonstrate the applicability of CORNETO across different FBA approaches:

(1) Simulation: We extend the sFBA method to identify minimal metabolic networks that retain essential functional capabilities, such as biomass production. This allows simulation of metabolic states under hypothetical or constrained conditions.
(2) Prediction: We estimate metabolic fluxes from sparse and noisy measurements in multiple samples. By integrating experimental data, this approach constrains flux distributions to biologically feasible solutions, improving the accuracy of flux predictions under sample-limited conditions.

Multi-sample sFBA. sFBA[30,31] identifies core metabolic networks sustaining specific functions. Instead of maximizing a target flux (such as biomass), sFBA makes this a constraint and minimizes the number of active reactions, selecting the smallest reaction set fulfilling the metabolic constraints.

However, standard sFBA cannot consider multiple samples simultaneously, analysing each independently[32]. This overlooks shared pathways used across conditions. Furthermore, the underdetermined nature of FBA yields multiple equivalent optimal solutions[33,34], obscuring shared strategies and potentially inflating differences in downstream analyses.

CORNETO's multi-sample sFBA simultaneously optimizes networks across all samples, minimizing the union of the inferred networks while satisfying sample-specific constraints (for example, nutrients and bounds). Unlike standard FBA, each sample has its own flux vector. Sparsity is applied globally, minimizing the total number of distinct reactions used across all samples, rather than per sample. This avoids overestimating differences and implicitly captures shared pathways. As flux optimization is a constraint, the distinction is between $\lambda = 0$ (no sparsity) and $\lambda > 0$ (sparsity applied).

To test performance, we used the MitoCore metabolic network[35] (555 reactions, 441 metabolites), simulating single reaction knockouts as different samples (Supplementary Information).

We compared single sFBA and CORNETO multi-sample sFBA, enforcing biomass production at ≥10%, 50% and 90% of the optimal level for each knockout (Fig. 3a).

CORNETO's multi-sample sFBA consistently produces networks with fewer unique reactions than standard sFBA. Median network size reduction was 6.02% (at ≥90% optimal biomass), 7.21% (≥50%) and up to 30% (at ≥10%). These differences reflect how the flexibility of flux distributions decreases as the biomass optimality threshold increases: requiring solutions to approach the theoretical optimum tightly constrains the feasible space, leaving fewer alternative pathways and, therefore, fewer differences between the single- or multi-sample sFBA solution.

At the ≥90% biomass threshold, multi-sample sFBA more frequently selects key glycolysis reactions, for example, hexokinase 1 (HEX1), glucose-6-phosphate isomerase (PGI) and phosphofructokinase (PFK) (Fig. 3b). This preference for canonical glycolytic reactions aligns with the minimal pathway necessary for biomass production[36].

While single sFBA might identify alternative pathways, multi-sample sFBA consistently selects more canonical glycolytic reactions across knockouts. This suggests that glycolysis is a more robust minimal route than indicated by single-sample analysis, which has a higher risk of selecting alternative pathways and exaggerating intersample differences.

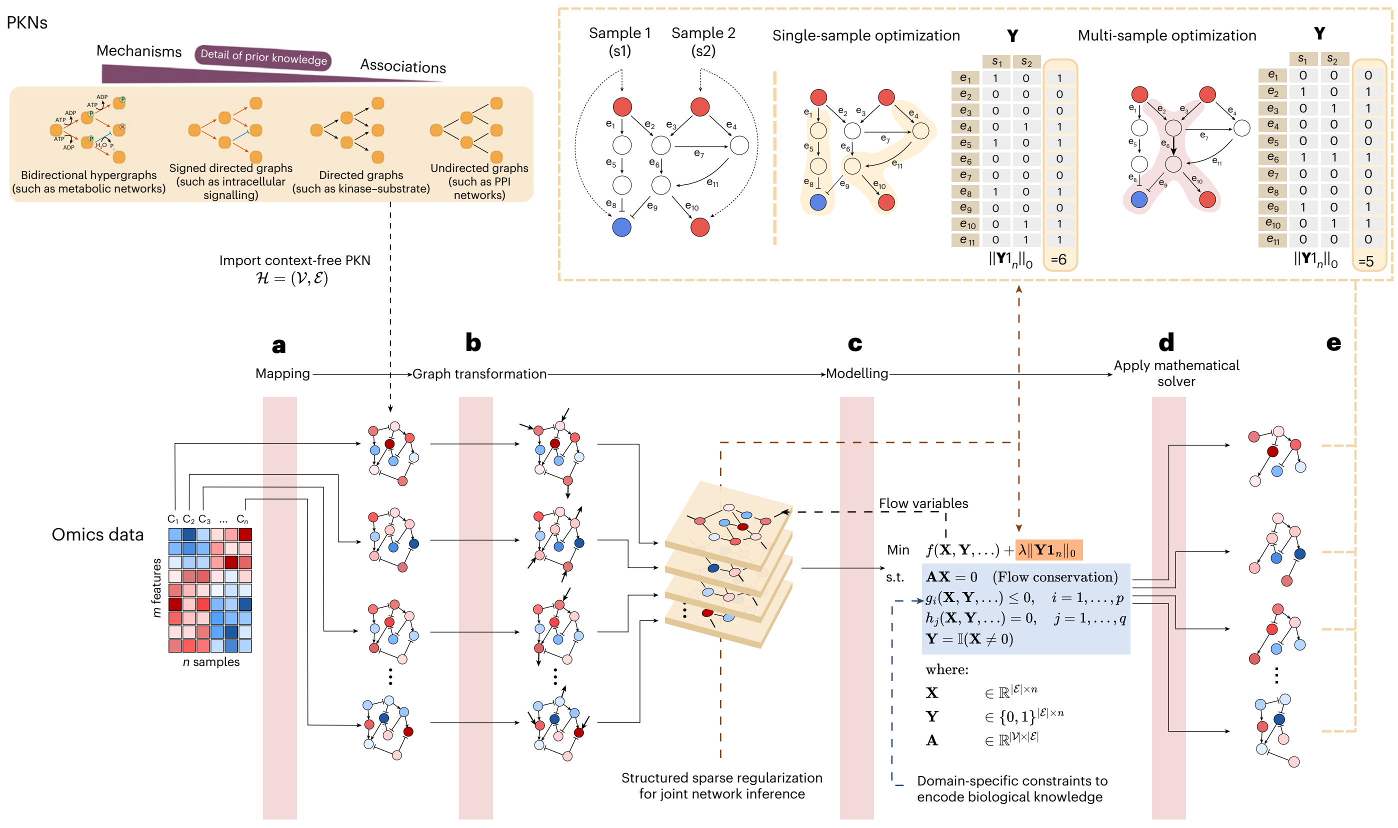
PKNs
Mechanisms
Detail of prior knowledge
Associations
Bidirectional hypergraphs (such as metabolic networks)
Signed directed graphs (such as intracellular signalling)
Directed graphs (such as kinase–substrate)
Undirected graphs (such as PPI networks)
Import context-free PKN $\mathcal{H} = (\mathcal{V}, \mathcal{E})$
Sample 1 (s1)
Sample 2 (s2)
Single-sample optimization
Multi-sample optimization
Y
$\|\mathbf{Y}1_n\|_0$ =6
$\|\mathbf{Y}1_n\|_0$ =5
a Mapping
b Graph transformation
c Modelling
d Apply mathematical solver
e
Omics data
m features
n samples
Flow variables
Min $f(\mathbf{X}, \mathbf{Y}, \ldots) + \lambda\|\mathbf{Y}1_n\|_0$
s.t. $\mathbf{AX} = 0$ (Flow conservation)
$g_i(\mathbf{X}, \mathbf{Y}, \ldots) \leq 0, \quad i = 1, \ldots, p$
$h_j(\mathbf{X}, \mathbf{Y}, \ldots) = 0, \quad j = 1, \ldots, q$
$\mathbf{Y} = \mathbb{I}(\mathbf{X} \neq 0)$
where:
$\mathbf{X} \in \mathbb{R}^{|\mathcal{E}| \times n}$
$\mathbf{Y} \in \{0,1\}^{|\mathcal{E}| \times n}$
$\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$
Structured sparse regularization for joint network inference
Domain-specific constraints to encode biological knowledge

Fig. 2 | CORNETO unifies network inference methods from prior knowledge and omics, enabling multi-sample network inference through the lens of network flows and mixed-integer optimization. The framework operates on a context-free PKN $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ that contains known biological interactions and additional knowledge such as stoichiometric information, gene–protein reaction rules, type of interaction and so on, depending on the type of prior knowledge used. The framework supports multiple types of prior knowledge, including undirected, directed, signed directed graphs and hypergraphs. These graphs can be obtained from databases or be defined by the user. Every network inference method in CORNETO is implemented in terms of data mapping, graph transformations and conversion into a constrained optimization problem using mathematical programming. **a**, Omics data are transformed and mapped onto the context-free PKN, resulting in *n* annotated graphs, one per sample or condition. **b**, Graphs are transformed by the method to optimize the structure based on the input data and to adapt the graph to be suitable for converting it into an optimization problem using network flows. **c**, The network inference method is formulated as a constrained optimization problem, using flow and indicator variables created by CORNETO on top of the transformed graphs. The regularization term (orange) is added to the objective function to promote structured sparsity across all samples, enabling joint inference. **d**, The optimization problem is solved by one of the many mathematical solvers supported by CORNETO. This results in *n* optimal inferred networks, with proofs of optimality provided by the solvers. **e**, An example of a signalling network with two samples, where we use only information about receptors (top vertices) and TFs (bottom nodes), to infer the networks propagating the signal. Red colour indicates activation and blue inhibition. Single-sample optimization leads to two networks not sharing any edge, where the size (number of edges) of the union is six, whereas the joint (multi-sample) optimization finds two networks that share one edge, leading to a solution with only five edges from the PKN that correctly explains the observed activations and inhibitions.

In summary, CORNETO supports extending FBA-based methods, as demonstrated by implementing multi-sample sFBA. This method provides an optimal sFBA extension for joint multi-sample analysis. It is optimal in the sense that it guarantees finding the smallest network fulfilling all constraints across samples, if one exists.

Multi-sample metabolic flux adjustment. Next, we tested whether CORNETO can be extended to predict unobserved metabolic fluxes from sparse and noisy measurements. We developed a multi-sample metabolic flux adjustment method analogous to sFBA but based on ℓ_2-regularized flux adjustment, subject to the FBA constraints and the same global structured regularization penalty controlled by λ. To evaluate our approach, we sampled steady-state metabolic fluxes from the MitoCore metabolic network and systematically introduced missing values (Supplementary Information).

In the noiseless setting with only 20% of missing values, the test root mean square error (RMSE) remains extremely low, around 1×10^{-8}, indicating near-perfect reconstruction of the metabolic fluxes (Fig. 3c). This difference drastically increases as more missing values are introduced, but the model maintains a test RMSE of 1×10^{-8} for $\lambda > 0$ in the noiseless setting, suggesting that structured regularization effectively constrains the solution space even with substantial data loss. Adding noise decreases the accuracy of flux predictions, leading to higher RMSE values; however, there is a consistent advantage in using structured regularization, with test error still lower for noise levels up to 1.0. This advantage is particularly evident when 50% or more of the flux measurements are missing, where the global regularization term helps to stabilize predictions and prevent overfitting to noisy observations (Supplementary Fig. 3).

These results highlight that leveraging structured regularization in a targeted manner enables more accurate flux predictions with reduced data requirements, reinforcing the notion that sparser models can outperform large, comprehensive networks in certain contexts, as has been demonstrated by other methods[37–39].

CORNETO enables multi-sample inference of context-specific metabolic networks from omics data

Integrating omics data enhances inference of context-specific metabolic networks. Techniques like iMAT[27], FastCORE[40] or GIMME[41] map omics measurements onto metabolic networks and optimize a score

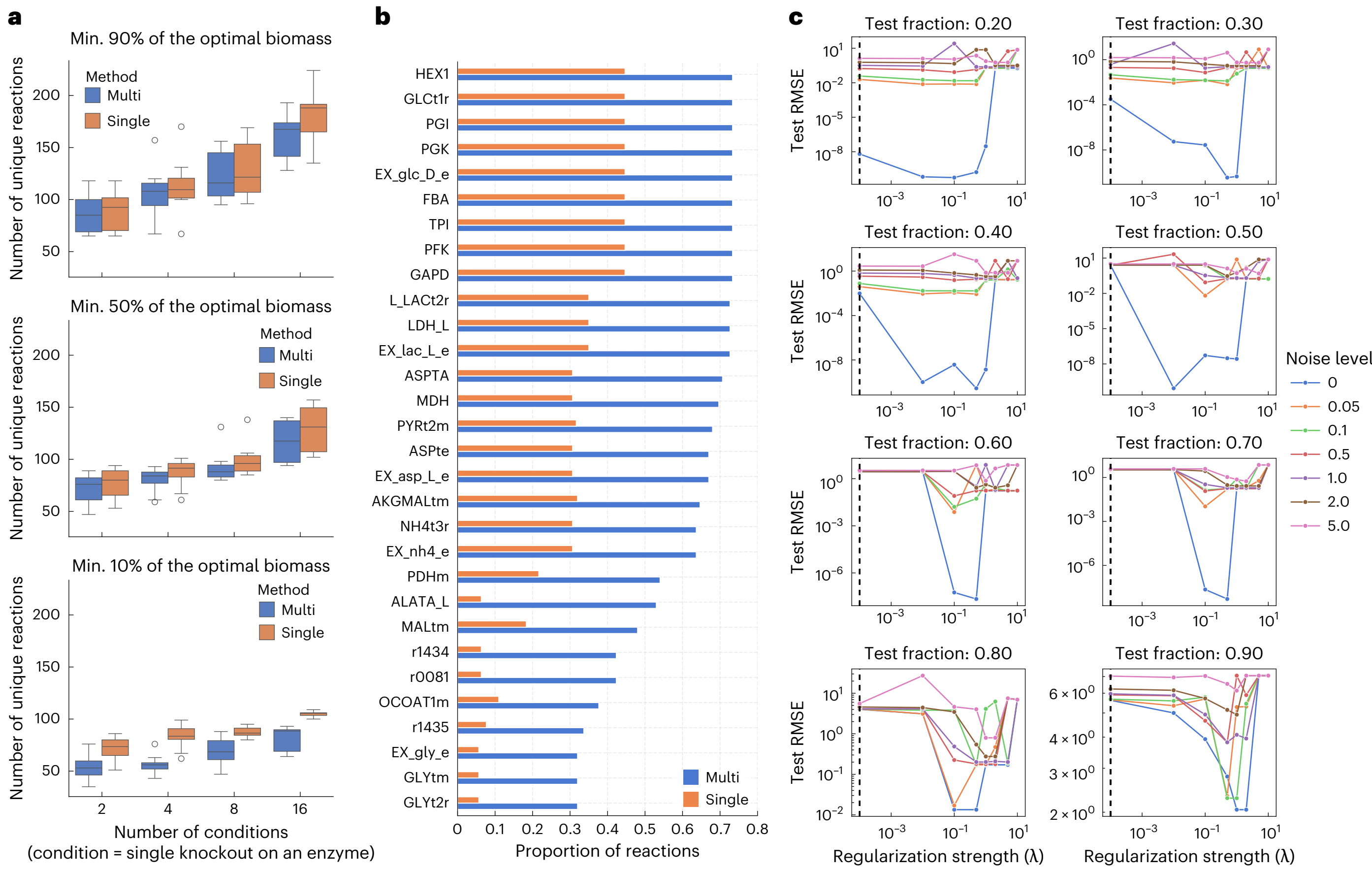

Fig. 3 | Simulation study using CORNETO's capabilities for multi-sample FBA modelling. a, The number of different reactions selected using the single-sample sFBA (orange) versus CORNETO's multi-sample sFBA (blue), constraining the optimal production of biomass to be a minimum (min.) of 90% (top), 50% (middle) and 10% (bottom) of the optimal attainable biomass for each sample, and for four different scenarios considering 2, 4, 8 and 16 samples. Each sample represents a single, simulated knockout (condition). For each scenario, we generated ten independent subsets of the specified size by randomly sampling conditions from the full pool of knockouts ($n = 10$ per boxplot). **b**, The average selection proportion of each reaction across different sample sizes (2, 4, 8 and 16 samples), where 1 indicates selection of the reaction in every case. Results are shown for the top 30 reactions with the greatest differences in selection frequency between the multi-sFBA and single-sFBA methods, under the minimum 90% biomass constraint setting. **c**, The performance of the sparse flux adjustment method leaving out different percentages of metabolic fluxes across samples, and under different Gaussian noise, for different regularization strengths (λ). The vertical dashed line indicates the baseline results for $\lambda = 0$.

function maximizing selection of context-specific reactions, subject to metabolic constraints[42]. Similar to sFBA and other FBA-based methodologies, these methods infer networks using single samples.

We used CORNETO to develop a multi-sample context-specific metabolic network reconstruction method from omics data, using an objective function similar to iMAT. In iMAT, reaction scores from transcriptomics data guide optimal subnetwork selection by balancing inclusion of positive-score reactions against exclusion of negative-score ones. A threshold classifies enzyme-coding genes as highly or lowly expressed.

We applied it to a yeast gene deletion microarray dataset[43]. To ensure sufficient signal, we initially excluded gene knockouts not encoding enzymes. We focused on knockouts causing significant expression changes in other enzyme-encoding genes (Supplementary Information). We identified 12 enzyme-encoding gene knockouts suggesting potential metabolic rewiring (Supplementary Fig. 4).

To determine the optimal λ for multi-sample iMAT, we performed fivefold cross-validation[27]. In each fold, 20% of reaction scores were held out for validation and 80% used for inference. We repeated this for different λ values to estimate generalization errors and identify the best parameter. Based on the F1 score, best performance is with $\lambda = 1.5$, yielding a 6.62% average F1-score improvement (Fig. 4a) compared with baseline ($\lambda = 0$, original single-condition iMAT). This translates to a 29.62% precision increase (Fig. 4b), at the cost of a 8.62% recall reduction (Fig. 4c). The network size decreases from ~2,000 different reactions to 860 distinct reactions on average (Fig. 4d), and total false positives decrease by 38.80% (Supplementary Fig. 5a).

We compared selected reactions using the original iMAT and CORNETO's multi-sample method ($\lambda = 1.5$). The multi-sample approach reduced differences across samples by selecting more unique reactions in fatty acid degradation and fewer in glycerolipid metabolism (Fig. 4e). These pathways contain a high proportion of reactions lacking gene annotations, resulting in network regions that are unconstrained by transcriptomic data. This allows greater flexibility in the selection of alternative flux routes during optimization, as these reactions can be freely included or excluded without affecting the fit to gene expression-derived scores.

CORNETO's multi-sample iMAT method includes more fatty acid degradation reactions under specific knockouts (*PFK2*, *ACO1*, *PLC1*, *IPK1*, *RNR4* and *ARG82* genes) (Fig. 4f). A closer examination of the input data reveals that, for these knockouts, the corresponding enzyme-coding genes were not differentially regulated (score of 0), whereas in other samples these reactions predominantly did (positive scores; Supplementary Fig. 5b). This observation highlights that the multi-sample approach leverages all available data when deciding which reactions to include on a per-sample basis. Here, the method

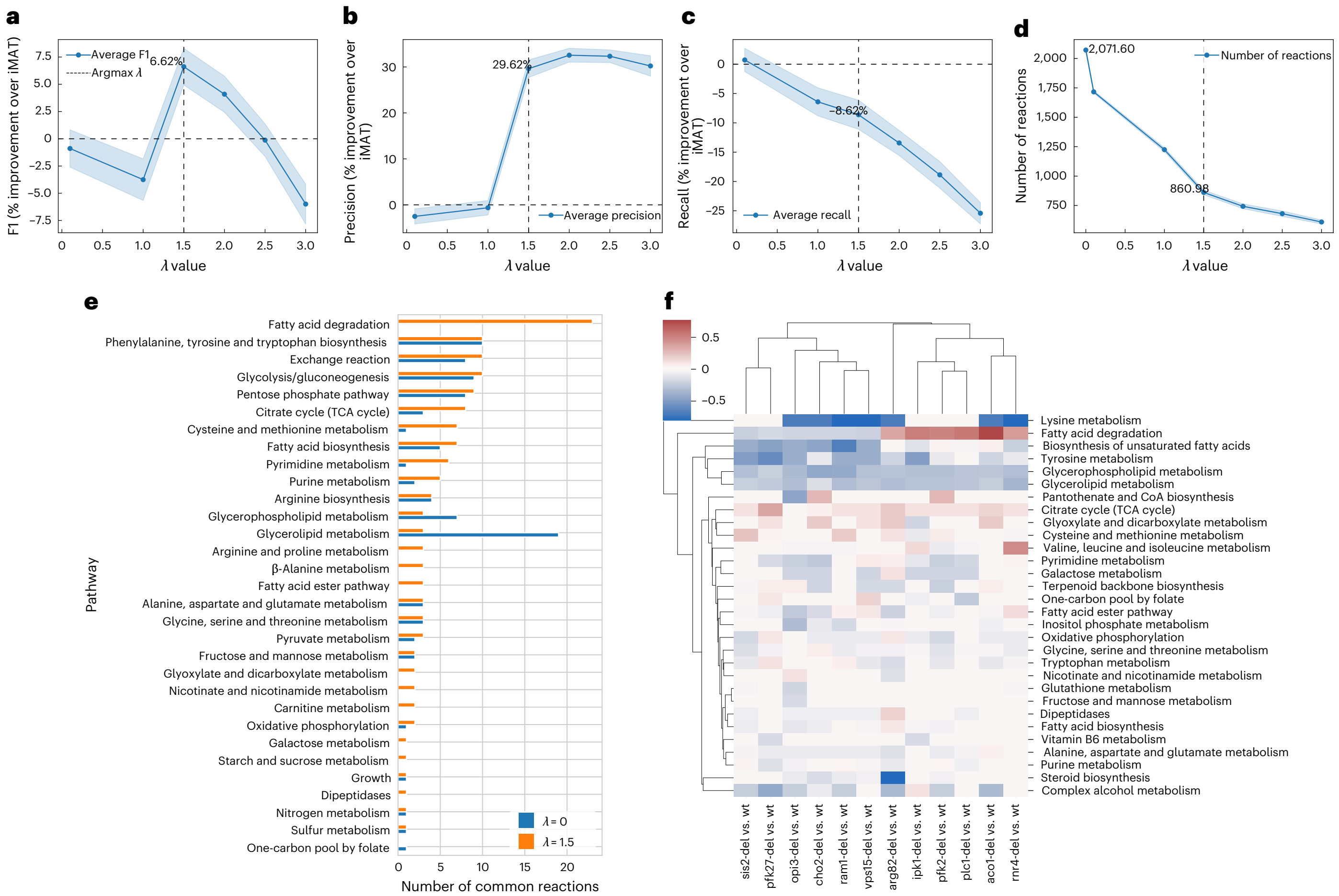

Fig. 4 | Evaluation and comparison of single- and multi-sample metabolic context-specific network inference. a, The average percentage improvement in F1-score over iMAT (single sample for different λ values on a fivefold cross-validation setting). The vertical dashed line is the optimal selected λ using the F1 score. Error bands indicate $\pm 1\sigma$ (standard deviation), over ten independent runs with different random seeds. **b**, Improvement of precision relative to iMAT. **c**, Decrease in recall relative to iMAT. **d**, The average number of different reactions selected by each method. **e**, The distribution across metabolic pathways of reactions common to all samples, as selected by single iMAT ($\lambda = 0$) and CORNETO's multi-sample iMAT ($\lambda = 1.5$). **f**, A heatmap of difference in proportion of selected reactions per pathway between multi-sample iMAT and single iMAT (red, more reactions selected by multi-sample iMAT; blue, more reactions selected by single iMAT).

recognizes that flux can be rerouted through the fatty acid degradation pathway without compromising the fitting error for these samples and, given the uncertain activity of these reactions, it favours their inclusion to predict a similar flux distribution as in the other samples.

Our analysis demonstrates the multi-sample method substantially enhances precision by reducing false positives in context-specific metabolic models.

Multi-patient signalling networks identify a higher proportion of interactions driven by shared deregulated kinases

Using CORNETO, we reimplemented and extended CARNIVAL[28] to handle multi-sample settings. CARNIVAL is a method that infers signalling networks by analysing transcription factor (TF) activity, estimated from transcriptomics data and known TF targets[44], and reconstructs networks from known inputs (for example, drug targets or receptors) to TFs. This links input perturbations to TFs via the PKN, accounting for interaction directionality and sign (activation or inhibition), allowing the study of signal flow. If perturbations are unknown, all potential upstream vertices in the PKN with no input edges are selected as potential sources for signal propagation—an approach we refer to as inverse CARNIVAL.

We first assessed performance using synthetic ground-truth networks generated from transcriptomics data of drug-treated cell lines under different regularization settings, enforcing varying degrees of similarity in the inferred signalling networks. As expected, precision increases when the true networks exhibit greater similarity across samples (Supplementary Fig. 6).

We then applied CORNETO to infer patient-specific intracellular networks from transcriptomic data from patients with lung adenocarcinoma profiled by the Clinical Proteomic Tumor Analysis Consortium (CPTAC)[45,46], leveraging phosphoproteomics data for validation (Supplementary Information).

For validation, we defined shared deregulated kinases as those with absolute activity scores in the top quartile for at least 50% of the patients. We then compared the performance of the single-sample and multi-sample approaches by assessing both the total number of network edges (the union across patients) and the intersecting edges shared among patient networks (Fig. 5b). For $\lambda < 0.8$, the single-sample setting produced a higher total number of edges. However, the number of vertices remained similar for $\lambda > 0.5$, suggesting that multi-sample regularization enhances the consistency of edge detection without substantially altering network complexity.

The intersection of edges across patient networks was larger in the multi-sample setting, and a greater proportion of these edges were associated with the shared deregulated kinases. Moreover, within this intersection, a higher proportion of edges originated from the shared

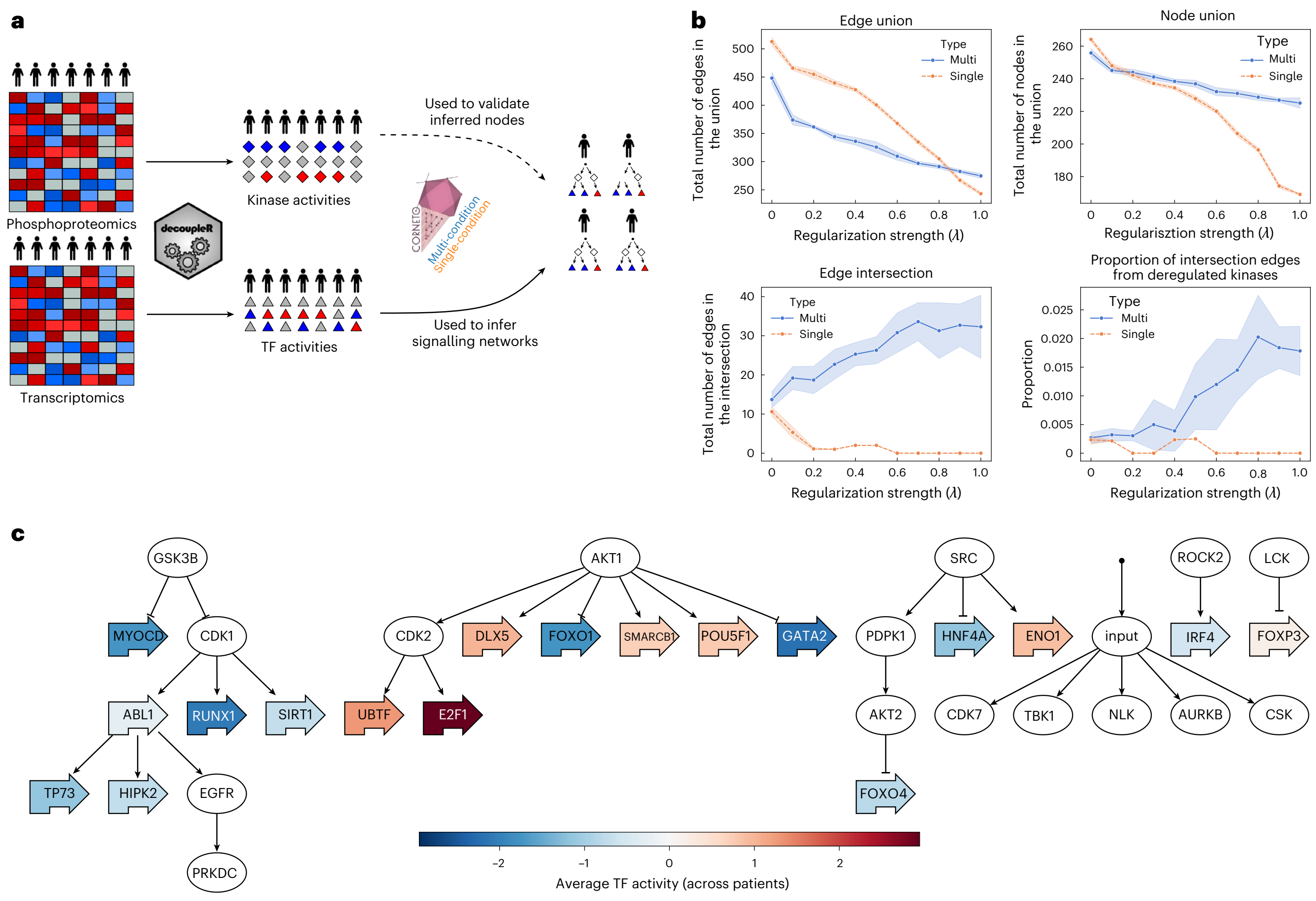

Fig. 5 | Evaluation and comparison of the single- and multi-sample intracellular signalling network inference on CPTAC. a, Overview of the approach. Using transcriptomics from patients, we estimated TF activities with Decoupler, which was used as an input for single-sample and multi-sample CARNIVAL. In both cases, we assume that the inputs (receptors) are unknown, and they are automatically selected during optimization (inverse CARNIVAL). For validation, we used phosphoproteomics data for the same patients, from which kinase activities were estimated. **b**, Results for the single- and multi-sample CARNIVAL, showing the average number of selected edges from the PKN for all patients (top left), same for the vertices (top right), the intersection of edges across patients (bottom left) and the mean proportion of correctly inferred interactions involving deregulated kinases, used as a validation set. Error bands indicate $\pm 1\sigma$ over ten independent runs (with different random seeds) at each value of λ. **c**, Interactions that appear in the intersection of the inferred networks for the optimal $\lambda = 0.8$. To account for variability, the multi-sample inference was repeated 30 times with different seeds, and we kept the interactions of the intersection if they appeared in at least 50% of the alternative solutions. The colours correspond to the average TF activity across patients.

deregulated kinases used as the validation set. This finding suggests an improved precision of the multi-sample approach in capturing biologically relevant kinase interactions, as these kinases were identified independently from the data used for inferring the networks. The intersection of patient-specific networks reveals shared signalling mechanisms, with key deregulated kinases such as GSK3B, AKT1 and SRC consistently influencing TF activity across patients (Fig. 5c).

Discussion

This Article presents CORNETO, a unified framework for network inference enabling multi-sample learning. By reformulating methods as joint optimization problems using mixed-integer optimization and network flows, CORNETO captures shared and context-specific features across samples. Its versatility makes CORNETO applicable to PPI, metabolic and signalling networks. We demonstrated its effectiveness across simulations and studies using real omics data.

To leverage multi-sample information, we introduced a structured sparsity-inducing penalty controlled by a λ parameter. This approach assumes that samples share similar network structures. For example, when studying multiple perturbations within the same organism or related subjects under the same perturbation (for example, different cell lines responding to the same treatment), network structures probably have shared edges. Larger λ values indirectly promote increased similarity, reducing variance and simplifying network structure. We showed this approach's effectiveness across different case studies, showing that implemented methods infer smaller, more consistent networks.

Using CORNETO, we implemented and extended multiple network inference methods based on mixed-integer optimization and network flows. Implemented methods included FBA, sFBA and iMAT (metabolism), CARNIVAL (intracellular signalling) and various Steiner trees and PCSTs for PPI networks. We illustrate CORNETO's versatility through multiple applications involving method extension or adaptation. Specifically, we applied CORNETO to (1) extend FBA to multi-sample scenarios; (2) integrate multi-sample omics data with FBA to derive flux-consistent, context-specific metabolic networks; and (3) jointly infer intracellular signalling networks from multi-sample transcriptomics data. In addition, we showcase CORNETO's broader applicability by adapting Steiner tree-based approaches for multi-sample protein interaction and kinase–substrate networks, and by optimizing biologically informed neural network architectures trained on single-cell transcriptomics, detailed in the Supplementary Information (Table 1).

A limitation of CORNETO is its scalability and computational complexity, because many methods involve non-deterministic polynomial-time hard (NP-hard) combinatorial problems. These demands grow exponentially as sample sizes or PKNs increase. Heuristic approaches, such as the Steiner-tree implementation in NetworkX or Fast PCST[47], provide faster solutions but generally target undirected networks, lack support for multiple samples or extra constraints and are suboptimal. Alternatively, heuristics could be used to find high-quality feasible solutions to initialize CORNETO's solvers for faster convergence. Exploring strategies such as Alternating Direction Method of Multipliers (ADMM) or relaxed formulations[48] could also enable efficient parallelization and distributed computing. As CORNETO is solver-agnostic (decouples the problem formulation from the optimization), it can in principle be adapted to leverage new optimization tools and techniques as they emerge.

Moreover, while CORNETO currently provides a flow conservation based framework for steady-state scenarios, extending it to dynamic or equilibrium settings is a promising future direction. Such extensions often involve time discretization[49]. Introducing dynamics brings challenges (increased complexity, uncertainty quantification, parameter fitting and sensitivity analysis), addressed in computational modelling literature[50,51]. Bridging steady-state and dynamic models remains an area for future research.

A promising future direction is extending methods to multi-layer network inference, integrating various PKNs and biological assumptions into a unified joint optimization approach. Its ability to import metabolic networks supported by COBRA (COnstraint-Based Reconstruction and Analysis Toolbox) makes it an ideal bridge for integrating metabolism with other cellular processes, developing approaches that incorporate diverse prior knowledge and data within a unified modelling framework. This could lead to models capturing interactions between biological processes, such as combining FBA with gene regulation[52–54], further integrating FBA with signalling and metabolic networks, extending beyond current approaches to causal signal-metabolic network integration[55] or enhancing models of cell–cell communication in spatial contexts[56]. These capabilities could aid in the development of digital twins[57] or virtual cell models[58,59] by leveraging different types of prior knowledge.

Beyond network inference, CORNETO's ability to flexibly model diverse biological mechanisms and incorporate additional constraints could make it well suited to inform experimental design. By simulating in silico perturbations in the network, or measuring variability under different constraints, the framework may help to prioritize interventions that most effectively constrain the solution space.

Another framework extension involves using CORNETO to derive biologically inspired artificial neural network architectures. We demonstrated that this is possible, showing how CORNETO can automatically extract context-specific networks from a PKN and convert them into a biologically inspired machine learning model, improving upon ref. 60. This approach could be extended to support constructing more advanced architectures, such as LEMBAS (Large-scale knowledge-EMBedded Artificial Signaling-networks) for intracellular signalling[61] or metabolism[62], in an automated and optimal fashion. CORNETO provides an automated way to generate context-specific networks that account for multiple samples, reducing the need for manual curation. Future work could explore integrating more complex architectures to enhance biological interpretability and predictive power. General frameworks such as Networks Commons[63] that allow swapping and combining network-inference algorithms, and linking them to diverse knowledge and data sources, will catalyse such developments.

In summary, CORNETO offers a unified framework for network inference, tailored for multi-sample analyses. By reformulating network inference problems as joint optimization tasks, CORNETO reduces variability across samples, increases precision, decreases the overall size of the inferred networks and improves the detection of shared biological interactions. Its flexibility integrating diverse PKNs makes it applicable across a wide range of biological contexts. The Python package makes these tools accessible, encouraging broader use and collaboration within the research community, paving the way for future innovations in network inference.

Methods

Mathematical background

Mixed-integer optimization. Mixed-integer programming is a type of constrained optimization problem that involves decision variables that can be both integer and continuous. A mixed-integer programming problem can be defined as follows:

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^n,\mathbf{y}\in\mathbb{Z}^m} \quad & f(\mathbf{x},\mathbf{y})\\ \text{subject to} \quad & g_i(\mathbf{x},\mathbf{y})\le 0, \quad i=1,2,\dots,k,\\ & h_j(\mathbf{x},\mathbf{y})=0, \quad j=1,2,\dots,l.\end{aligned}$$

where $f:\mathbb{R}^n\times\mathbb{Z}^m\to\mathbb{R}$ is the objective function, which involves both real and integer decision variables. The goal is to minimize (or maximize) this function, subject to inequality constraints $g_i(\mathbf{x},\mathbf{y})\le 0$, $i=1, 2, \dots, k$, where $g_i:\mathbb{R}^n\times\mathbb{Z}^m\to\mathbb{R}$, and equality constraints $h_j(\mathbf{x},\mathbf{y})=0$, $j=1,2,\dots,l$, where $h_j:\mathbb{R}^n\times\mathbb{Z}^m\to\mathbb{R}$.

If functions f, g_i and h_i are linear, the problem is a mixed-integer linear programming (MILP) problem. If all variables are continuous, the problem is a linear programming (LP) problem. Problems with integer variables are harder to solve as they are combinatorial in nature.

For these classes of problems, there exist many mathematical solvers that provide exact solutions with optimality guarantees, such as CPLEX, Gurobi, CBC, HIGHs and SCIP. We exploit the MILP representability of many network inference problems, allowing us to use these advanced solvers to find optimal solutions accurately.

Network flows. Network flows[23,24] are a well-studied and general class of problems that can be used to solve a wide variety of scientific and engineering problems. They model the distribution of quantities across a network, subject to constraints such as flow conservation and capacity constraints.

Given its versatility, different types of problems can be reduced to or formulated using network flows[64–66], for which efficient optimization methods have been developed. We build upon this idea to unify diverse network inference problems by reducing them to network flow-based problems. By leveraging this framework, we can extend existing methods to account for joint inference on multi-sample scenarios.

We first introduce the basic network flow problem on a directed graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of directed edges. Let $\mathbf{x}\in\mathbb{R}^{|\mathcal{E}|}$ be the vector of flows on the edges, $\mathbf{b}\in\mathbb{R}^{|\mathcal{V}|}$ be the vector of net flows at the vertices and $\mathbf{x}_{max}\in\mathbb{R}^{|\mathcal{E}|}$ be the vector of upper bounds (capacities) for the flows on the edges. Let $A\in\mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ be the vertex–edge incidence matrix of the graph $\mathcal{G}$ defined as

$$A_{ij}=\begin{cases}-1 & \text{if } v_i=\text{tail}(e_j),\\ 1 & \text{if } v_i=\text{head}(e_j),\\ 0 & \text{otherwise}.\end{cases} \tag{1}$$

where head and tail are functions that, given edge $e=(v_t,v_h)\in\mathcal{E}$, return the head ($v_h$) and the tail ($v_t$) of e.

The network flow problem aims to find the vector of flows $\mathbf{x}$ that satisfies the flow capacity constraints and flow conservation constraints. This can be expressed as a linear system

$$A\mathbf{x}=\mathbf{b}, \quad 0\le\mathbf{x}\le\mathbf{x}_{\max}, \tag{2}$$

where the vector **b** is defined as

$$\mathbf{b}_i = \begin{cases} -B & \text{if } v_i = s, \\ B & \text{if } v_i = t, \\ 0 & \text{otherwise,} \end{cases}$$

where s is the source vertex, t is the sink vertex and B is the total flow from the source to the sink. The vector **b** represents the net flow entering or leaving each vertex, and it is non-zero only for source and sink vertices, with sources having negative values (net outflow) and sinks having positive values (net inflow).

In typical network flow problems, such as the minimum-cost flow problem, we are interested in finding an optimal point of this set Ω with respect to a linear objective function

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Omega} f(\mathbf{x}),$$

where f is frequently defined as $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$, with **c** representing the cost per unit flow on each edge, and Ω is the feasible set of valid flows in the network, defined as

$$\Omega = \{\mathbf{x} \in \mathbb{R}^{|\mathcal{E}|} | A\mathbf{x} = \mathbf{b},\ 0 \le \mathbf{x} \le \mathbf{x}_{\max}\}.$$

As the flows are continuous values and the objective function and constraints are linear, the problem is an LP problem. LP problems are efficiently solved using algorithms such as the Simplex and Interior Point methods[67], leveraging duality theory to provide bounds on the optimal value. Many mathematical solvers incorporate these algorithms to provide accurate solutions even for large-scale problems.

The canonical form of the network flow problem as an LP can be expressed as the minimization of a linear function subject to (s.t.) linear constraints

$$\begin{aligned} \min.\ & \mathbf{c}^T\mathbf{x}, \\ \text{s.t.}\ & A\mathbf{x} = \mathbf{b}, \\ & 0 \le \mathbf{x} \le \mathbf{x}_{\max}. \end{aligned}$$

More complex network problems can be modelled by extending this simple network flow model. In particular, network inference problems, in which the goal is not only to find the values of certain parameters but also to determine the structure of the network, can be modelled by incorporating binary indicator variables to decide whether a certain edge is included in the solution. For example, let $\mathbf{y} \in \{0,1\}^{|\mathcal{E}|}$ be a binary vector where $y_j = 1$ if edge e_j is included in the network and $y_j = 0$ otherwise. This extension transforms the problem into a MILP, which can be formulated as follows:

$$\begin{aligned} \min.\ & \mathbf{c}^T\mathbf{x} + \mathbf{d}^T\mathbf{y}, \\ \text{subject to}\ & A\mathbf{x} = \mathbf{b}, \\ & 0 \le \mathbf{x} \le \mathbf{x}_{\max} \odot \mathbf{y}, \\ & \mathbf{y} \in \{0,1\}^{|\mathcal{E}|}, \end{aligned}$$

where **d** represents the fixed costs associated with including each edge, and $\odot$ denotes hadamard (element-wise) product. The constraint $0 \le \mathbf{x} \le \mathbf{x}_{\max} \odot \mathbf{y}$ ensures that flow can only occur on edges that are included in the network.

The optimization problem represented by this MILP formulation is harder to solve than the standard LP problem owing to the presence of binary decision variables. However, it is more flexible and allows us to model a wide variety of problems that involve searching for the right combinatorial structures, such as paths, trees, acyclic graphs or other types of structures in an exact way. Moreover, MILP solvers are able to provide solutions along with certificates of optimality, ensuring that the solutions meet the user's defined criteria.

Model

In this section, we present the core model behind CORNETO. Our framework redefines network inference by transforming it into a network flow-based problem, leveraging the mixed-integer optimization framework commonly used in such problems. This unified approach enables joint inference by utilizing multiple samples simultaneously, allowing the model to improve network inference by borrowing information across all samples. In addition, our framework offers several key advantages: (1) all methods are implemented consistently with a common vocabulary and API; (2) we identify and reuse common components across different methods, streamlining development; and (3) our framework provides exact algorithm implementations, allowing solvers to find optimal or suboptimal solutions with certificates of optimality, giving users control over the solution quality.

Hypergraphs. To support a wide variety of methods and PKNs, CORNETO operates on directed hypergraphs. A directed hypergraph $\mathcal{H}$ is defined as $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices, and $\mathcal{E}$ is a set of directed hyperedges. Each hyperedge $e_i \in \mathcal{E}$ is an ordered pair (T_i, H_i), with $T_i \subseteq \mathcal{V}$ as the tail and $H_i \subseteq \mathcal{V}$ as the head of the hyperedge.

To accurately model diverse network interactions, particularly those requiring signed coefficients such as stoichiometry in metabolic networks, the vertex–edge incidence matrix for hypergraphs is defined on the basis of these direct coefficients.

Let $c_{ij} \in \mathbb{R}$ be a coefficient that quantifies the specific, signed involvement of vertex v_i in hyperedge e_j. The sign of c_{ij} indicates the nature of the participation of v_i in e_j (for example, as a reactant or product, source or target of influence), while its magnitude represents the extent of this participation.

The vertex–edge incidence matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ is defined as

$$\mathbf{A}_{ij} = \begin{cases} c_{ij} & \text{if vertex } v_i \text{ is in hyperedge } e_j, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

For a hyperedge $e_j = (T_j, H_j)$, the sets T_j (tail) and H_j (head) define the structural components of the hyperedge. When considering a 'forward' flow or process through e_j (that is, when the corresponding flow variable $x_j > 0$):

- Vertices $v_i \in T_j$ typically correspond to inputs or reactants, and their coefficients c_{ij} would often be negative (for example $c_{ij} < 0$).
- Vertices $v_i \in H_j$ typically correspond to outputs or products, and their coefficients c_{ij} would often be positive (for example $c_{ij} > 0$).

For example, for a simple directed-graph edge $e_j = (v_t, v_h)$ representing flow from v_t to v_h, the typical coefficients would be $c_{v_t,j} = -1$ and $c_{v_h,j} = 1$. All other $c_{v_k,j} = 0$ for this edge.

For a metabolic reaction e_j such as $M_1 + 2M_2 \rightarrow M_3$, where $T_j = \{M_1, M_2\}$ and $H_j = \{M_3\}$, the coefficients would be $c_{M_1,j} = -1$, $c_{M_2,j} = -2$ and $c_{M_3,j} = 1$.

This definition allows the flow-conservation constraint $\mathbf{Ax} = \mathbf{0}$ (where **x** is the vector of flows through hyperedges) to correctly represent complex balances, such as stoichiometric steady states in metabolic networks. A negative flow x_j for a hyperedge e_j would then signify that the process occurs in the reverse direction relative to the signs defined by the c_{ij} coefficients, as is commonly done in FBA.

In this framework, a column of **A** (representing one hyperedge) can contain multiple non-zero entries, with varying signs and magnitudes. This directly reflects hyperedges with multiple vertices in their tail and head sets, enabling the representation of complex relationships such as biochemical reactions (for example $A + B \rightleftharpoons C + D$) as well as or other multi-component processes. This differs from standard graphs, where each column in **A** (representing an edge) typically has exactly one −1 and one +1 entry (or is all zeros). In this way, the incidence matrix $\mathbf{A}$ plays a role directly analogous to the stoichiometric matrix in FBA when applied to metabolic networks.

Because a standard directed graph can be viewed as a particular instance of a directed hypergraph, in the following we will use $\mathcal{H}$ to denote hypergraphs (including graphs as a special case) and $\mathcal{G}$ to denote only graphs, when specific methods are not designed to operate on hypergraphs.

Data mapping function. Given a dataset $\mathbf{D} \in \mathbb{R}^{m\times n}$ (m features and n samples, such as genes and cells), and a PKN $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the function $\phi : (\mathcal{H}, \mathbf{D}) \to \mathcal{H}'$ is defined as

$$\phi : (\mathbf{D}, \mathcal{H}) \to \mathcal{H}' = (\mathcal{V}', \mathcal{E}'), \tag{4}$$

where

- $\mathcal{V}' = \{v' \mid v' = (v, \mathbf{d}_v)\}$ is the new set of vertices. Each vertex $v' = (v, \mathbf{d}_v)$ is associated with a vector $\mathbf{d}_v \in \mathbb{R}^n$, which contains the computed data for that vertex across all samples in the dataset. Specifically,

$$\mathbf{d}_v = \left[d_v^{(1)}, d_v^{(2)}, \ldots, d_v^{(n)}\right],$$

where $d_v^{(i)} \in \mathbb{R}$ is the computed feature value for vertex v in the ith sample.

- $\mathcal{E}' = \{e' \mid e' = (e, \mathbf{d}_e)\}$ is the new set of edges. Each edge $e' = (e, \mathbf{d}_e)$ is associated with a vector $\mathbf{d}_e \in \mathbb{R}^n$, which contains the computed values related to the interaction between the two vertices connected by edge e across all samples. Specifically,

$$\mathbf{d}_e = \left[d_e^{(1)}, d_e^{(2)}, \ldots, d_e^{(n)}\right],$$

where $d_e^{(i)} \in \mathbb{R}$ represents the computed value associated with the edge e for the ith sample.

The data mapping function generalizes the process of transforming omics data into network features by applying specific rules or transformations that align with a PKN. For example, in metabolic networks, gene–protein reaction rules are used to calculate reaction values (edges) based on the expression levels of specific genes from the dataset. Other operations might involve simpler transformations, such as discretization of values, where, for example, gene expression or protein levels are converted into binary states to represent activation or inhibition.

Graph transformation function. This function takes an annotated graph $\mathcal{H}'$ as input and returns a new graph by modifying its structure, such as by adding or removing vertices or edges.

Given an annotated graph $\mathcal{H}' = (\mathcal{V}', \mathcal{E}')$, we define the graph transformation function $\psi : \mathcal{H}' \to \mathcal{H}''$ as

$$\psi(\mathcal{H}') = \mathcal{H}'' = (\mathcal{V}'' = (\mathcal{V}' \setminus \mathcal{V}_r) \cup \mathcal{V}_a, \mathcal{E}'' = (\mathcal{E}' \setminus \mathcal{E}_r) \cup \mathcal{E}_a).$$

The function ψ modifies $\mathcal{H}'$ by removing vertices in the set $\mathcal{V}_r$ and edges in $\mathcal{E}_r$, and by adding vertices from $\mathcal{V}_a$ and edges from $\mathcal{E}_a$, resulting in the new vertex set $\mathcal{V}''$ and edge set $\mathcal{E}''$. These transformations are used to adapt the structure of a given graph to be compatible with a network flow problems and to implement optimization steps to reduce the complexity of the methods.

Each method in CORNETO provides a specific implementation of ψ by specifying how the vertices and edges are added or removed to align with its reformulation based on network flows.

Network flows on bidirectional hypergraphs. We extend the basic network flow problem (equation (2)) to support the modelling of network inference methods on hypergraphs. This extension allows us to incorporate complex biological prior knowledge that is not compatible with simple graph representations[68].

Instead of using special types of vertices s and t to maintain flow balance, as in the traditional source–sink model, we extend the concept by adding hyperedges with no tail to inject flow and no head to extract flow. Specifically, to inject a certain amount of flow into a vertex v, it suffices to add a hyperedge $e_{\text{source}} = (\varnothing, \{v\})$, which has an empty tail and the vertex v in its head, with the flow value set to the desired amount (for example, 1 unit). Similarly, to extract flow from a vertex v, a hyperedge $e_{\text{sink}} = (\{v\}, \varnothing)$ can be added to the graph, which has the vertex v in its tail and an empty head. Multiple edges to inject and remove flow can be added to the network.

The feasible set of flows is defined by the constraints on flow conservation and capacity limits, given by

$$\Omega = \{\mathbf{x} \in \mathbb{R}^{|\mathcal{E}|} | \mathbf{A}\mathbf{x} = 0,\ \mathbf{x}_{\min} \le \mathbf{x} \le \mathbf{x}_{\max}\}, \tag{5}$$

where $\mathbf{x}_{\min}$ represents the lower bounds on the flow values, and $\mathbf{x}_{\max}$ represents the upper bounds. If $\mathbf{x}_{\min}$ contains negative values, it indicates the possibility of negative flows, meaning that the flow can traverse the edge in the opposite direction. This modification allows bidirectional flows on hyperedges, controlled by the specified upper and lower bounds.

In this formulation, the vector $\mathbf{b}$ (which represents external flow demands) is implicitly incorporated into the structure of the hyperedges. By embedding the flow injection at vertex v and extraction at vertex v through special hyperedges e_{source} and e_{sink}, the external demands $\mathbf{b}$ are integrated into the incidence matrix A of the extended hypergraph. This transformation allows the flow problem to be represented as a homogeneous system $A\mathbf{x} = 0$, where the flow balance constraints are maintained naturally through the hypergraph structure.

The reformulation of methods in CORNETO's framework involve the use of graph transformations to manipulate the structure of the graph and to convert the original problem into a network flow-based problem. As a simple example, the shortest path problem from vertex s to vertex t can be reduced to a min-flow problem[69] by adding an edge to inject flow through s and extracting the same amount of flow through t and then minimizing the sum of the cost of each edge multiplied by the flow (Supplementary Fig. 2).

Multi-flows on bidirectional hypergraphs. A key feature of CORNETO is its ability to model problems involving multiple samples. This capability requires not only learning individual networks for each sample but also linking flows across these networks to support joint inference. Building upon the previously defined concepts for single network flows, we now extend the notion to multi-flows on bidirectional hypergraphs. This approach allows us to handle multiple types of flow simultaneously, modelling different samples within the same framework.

Similar to multi-commodity network flows[70], which involve multiple resources (commodities) moving through a network simultaneously with shared constraints, we extend the formulation to support multiple simultaneous flows. In the context of the framework, each sample or condition is associated with a flow, and each flow can be subjected to different constraints.

Let n be the number of samples (commodities). The flow vector $\mathbf{x}$ previously defined is now extended to a flow matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{E}|\times n}$, where $\mathbf{x}_{ij}$ represents the flow of sample j along hyperedge e_i. Similarly, we define $\mathbf{X}_{\min}, \mathbf{X}_{\max} \in \mathbb{R}^{|\mathcal{E}|\times n}$ to be the lower and upper bounds for the flow through the edges, for each sample.

Given $\mathbf{X}$, $\mathbf{X}_{\min}$ and $\mathbf{X}_{\max}$, the multi-flow problem on bidirected hypergraphs is given by

$$\mathbf{A}\mathbf{X} = 0$$

$$\mathbf{X}_{\min} \le \mathbf{X} \le \mathbf{X}_{\max}.$$

This formulation, however, treats each sample as an independent network flow problem. We will use this as a building block to define other problems, starting from this multi-flow problem. To model interconnected flows, we can introduce additional transformations and constraints on $\mathbf{X}$.

As a simple example, in a multi-commodity problem, instead of having independent edge capacities per commodity, a single capacity constraint for each edge is imposed. The sum of absolute flows across all commodities cannot exceed these shared bounds. Given shared bounds $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ for all commodities, we transform the multi-flow problem into a multi-commodity flow problem by summing the total flows for each edge:

$$\mathbf{AX} = 0$$
$$\mathbf{x}_{\min} \leq |\mathbf{X}|\mathbf{1}_n \leq \mathbf{x}_{\max}.$$

Here, the term $|\mathbf{X}|$ represents the absolute value of the flow matrix, ensuring that the sum of flows accounts for any potential negative values. The vector $\mathbf{1}_n$ is a column vector of n ones. The multiplication computes the sum of flows across all commodities.

To handle the nonlinearity introduced by the absolute value $|\mathbf{X}|$, it is possible to linearize it by introducing auxiliary variables for each x_{ij} in $\mathbf{X}$. This involves defining two non-negative variables, x_{ij}^+ and x_{ij}^-, such that $x_{ij} = x_{ij}^+ - x_{ij}^-$ and $|x_{ij}| = x_{ij}^+ + x_{ij}^-$. This transformation replaces the absolute value constraint with linear constraints, making the problem solvable using standard integer LP techniques.

Constrained optimization problems. In CORNETO, network inference methods are implemented as constrained optimization problems. An optimization problem is defined as a tuple:

$$\mathcal{P} = (f, \Omega), \tag{6}$$

where

- $f : \mathcal{X} \to \mathbb{R}$ is the objective function to be minimized, which operates on the decision variable space $\mathcal{X}$;
- Ω is the feasible set, defined by equalities and inequalities on the variables

$$\Omega = \left\{ x \in \mathcal{X} \,\middle|\, \begin{array}{ll} g_i(x) \leq 0, & i = 1, \ldots, k, \\ h_j(x) = 0, & j = 1, \ldots, l \end{array} \right\}.$$

We also define a trivial objective function as

$$f_0 : \mathcal{X} \to \mathbb{R}, \quad f_0(x) = 0 \quad \forall x \in \mathcal{X}. \tag{7}$$

In this case, (f_0, Ω) simply involves finding any feasible point in Ω.

All methods implemented in the framework use linear functions for f, g and h, over continuous and binary variables (with the exception of the multi-sample metabolic flux adjustment method, where the objective function is quadratic instead of linear).

Problem composition. Problems can be composed to define and reuse subproblems, thereby creating variations of existing methods without rewriting common pieces of code.

Let us consider two constrained optimization problems

$$\mathcal{P}_1 = (f_1, \Omega_1), \qquad \mathcal{P}_2 = (f_2, \Omega_2),$$

where

- f_1 and f_2 are objective functions to be minimized, and
- Ω_1 and Ω_2 are their respective feasible sets (each expressed in terms of the variables that appear in the corresponding problem).

We build the composed problem $\mathcal{P}_c = (f_c, \Omega_c)$ as follows:

- Combined objective. Choose non-negative weights α and β and set

$$f_c = \alpha f_1 + \beta f_2.$$

- Combined feasible set.

$$\Omega_c = \Omega_1 \cap \Omega_2,$$

that is, all constraints from $\mathcal{P}_1$ and $\mathcal{P}_2$ must hold simultaneously. Variables that appear in both original problems are implicitly identified; variables that appear in only one remain independent.

The composed optimization problem therefore reads

$$\min_{x \in \Omega_c} f_c(x).$$

To model, compose and solve constrained optimization problems in Python, CORNETO implements a multi-backend strategy that decouples the high-level modelling of network inference problems from the low-level canonical forms used by solvers. It currently supports CVXPY[71] and PICOS[72]. These backends also support sparse matrices and vectorization, which CORNETO leverages for efficient implementation of methods.

Building blocks. CORNETO defines a series of building blocks, composed by a set of variables and constraints, that can be used to construct subproblems. These subproblems can be combined to compose the feasible set of a given network inference method, representing the valid space of solutions with specific properties.

By combining these building blocks, it is possible to develop network inference methods without the need to reimplement common constraints, such as enforcing acyclicity on a graph. For example, Steiner tree problems explore the space of acyclic trees, and as such, we can easily build tree-based problems by imposing acyclicity on flows, which would serve as the foundation for all methods that infer tree-like networks. This approach not only simplifies the process of constructing new algorithms but also ensures that any improvements or extensions to the building blocks can be easily reused across multiple methods, allowing broader and more efficient enhancements without the need for redundant implementations.

Flow. Given a directed (hyper)graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, and upper and lower bounds for each edge in each sample $\mathbf{X}_{\min}, \mathbf{X}_{\max} \in \mathbb{R}^{|\mathcal{E}| \times n}$,

$$\Omega_{\text{Flow}}(\mathcal{H}) = \{\mathbf{X} | \mathbf{AX} = 0, \mathbf{X}_{\min} \leq \mathbf{X} \leq \mathbf{X}_{\max}\}, \tag{8}$$

where $\mathbf{X} \in \mathbb{R}^{|\mathcal{E}| \times n}$ is the variable representing the flows for each edge and sample. $\mathbf{A}$ is the vertex–edge incidence matrix of $\mathcal{H}$ (equation (1)).

Free variable indicator constraints. Given $\mathbf{X} \in \mathbb{R}^{m \times n}$ such that $\mathbf{X} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]$,

$$\Omega_{\text{Free}}(\mathbf{X}) = \{\mathbf{Y} | \mathbf{X}_{\min} \odot \mathbf{Y} \leq \mathbf{X} \leq \mathbf{X}_{\max} \odot \mathbf{Y}\}, \tag{9}$$

where $\mathbf{Y} \in \{0,1\}^{m \times n}$ is a binary variable such that $\mathbf{Y}_{i,j} = 0 \Rightarrow \mathbf{X}_{i,j} = 0$.

Non-zero indicator constraints. Given $\mathbf{X} \in \mathbb{R}^{m \times n}$, and $\mathbf{X} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]$,

$$\Omega_{\neq 0}(\mathbf{X}) = \left\{\mathbf{Y} \in \{0,1\}^{m \times n} \,|\, \mathbf{Y} \text{ satisfies (C1)} - \text{(C3)}\right\}, \tag{10}$$

where the constraints are defined as

$$\mathbf{Y} = \mathbf{Y}^+ + \mathbf{Y}^- \tag{C1}$$

$$\mathbf{Y} \leq 1 \tag{C2}$$

$$\mathbf{X}_{\min} \odot \mathbf{Y}^- + \epsilon \mathbf{Y}^+ \leq \mathbf{X} \leq \mathbf{X}_{\max} \odot \mathbf{Y}^+ - \epsilon \mathbf{Y}^-. \tag{C3}$$

Here, $\mathbf{Y}^+, \mathbf{Y}^- \in \{0,1\}^{m\times n}$ variables are introduced for linearizing the samples on $\mathbf{X}$, such that $\mathbf{Y}^+_{i,j} = 1 \iff \mathbf{X}_{i,j} \geq \epsilon$ (indicating that $\mathbf{X}$ is active in the positive direction if and only if $\mathbf{Y}^+_{i,j} = 1$), and $\mathbf{Y}^-_{i,j} = 1 \iff \mathbf{X}_{i,j} \leq -\epsilon$ (indicating that $\mathbf{X}$ is active in the negative direction if and only if $\mathbf{Y}^-_{i,j} = 1$).

Acyclic. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and the associated $\mathbf{X}$ flow variables,

$$\Omega_{\text{Acyc}}(\mathcal{G}) = \left\{\mathbf{Y} \in \Omega_{\neq 0}(\mathbf{X}) \mid \mathbf{Y} \text{ satisfies (C4)} - \text{(C6)}\right\}, \tag{11}$$

where the constraints are defined as

$$1 \leq \mathbf{L}_{v,:} \leq |\mathcal{V}|, \forall v \in \mathcal{V} \tag{C4}$$

$$\mathbf{L}_{v,:} - \mathbf{L}_{u,:} \geq \mathbf{Y}^+_{e,:} + (1 - |\mathcal{V}|)(1 - \mathbf{Y}^+_{e,:}) \tag{C5}$$

$$\mathbf{L}_{u,:} - \mathbf{L}_{v,:} \geq \mathbf{Y}^-_{e,:} + (1 - |\mathcal{V}|)(1 - \mathbf{Y}^-_{e,:}) \tag{C6}$$

for all edges $e = (u, v) \in \mathcal{E}$. Here, $\mathbf{L} \in \mathbb{R}^{|\mathcal{V}|\times n}$ is a continuous variable matrix that encodes a valid topological ordering of the vertices in the graph for each of the n samples. $\mathbf{Y} \in \{0,1\}^{|\mathcal{E}|\times n}$ is a non-zero binary indicator for the flows (equation (10)). We use u and v to refer to the indices of the source and target vertices of an edge in $\mathbf{L}$. The constraints are enforced for all columns (samples) in the matrices. Constraint (C4) ensures that, for each vertex v, its assigned order $\mathbf{L}_{v,:}$ is between 1 and $|\mathcal{V}|$. Constraints (C5) and (C6) enforce that, if an edge $e = (u, v) \in \mathcal{E}$ is directed from u to v (that is, $\mathbf{Y}^+_{e,:} = 1$), then the topological order of u must be less than that of v across all samples; specifically, $\mathbf{L}_{v,:} - \mathbf{L}_{u,:} \geq 1$. Similarly, if the edge is directed from v to u (that is, $\mathbf{Y}^-_{e,:} = 1$), then the order of v must be less than that of u; that is, $\mathbf{L}_{u,:} - \mathbf{L}_{v,:} \geq 1$. These conditions guarantee that the resulting topological ordering is consistent with an acyclic graph structure for all samples, depending on the directionality of the edges as defined by $\mathbf{Y}^+$ and $\mathbf{Y}^-$.

Note that the acyclicity constraint is specifically defined for graphs and not for hypergraphs. It is used in methods that aim to find directed acyclic graphs or tree structures within a PKN.

Tree. A tree is a directed acyclic graph where each vertex has at most one incoming edge. The tree constraint can be enforced by building on the acyclic structure $\Omega_{\text{Acyc}}(\mathcal{G})$ and adding a constraint that ensures each vertex $v \in \mathcal{V}$ has no more than one incoming edge across all samples. The constraint is formulated as follows:

$$\Omega_{\text{Tree}}(\mathcal{G}) = \left\{\mathbf{Y} \in \Omega_{\text{Acyc}}(\mathcal{G}) \;\middle|\; \sum_{e \in \text{In}(v)} \mathbf{Y}_{e,:} \leq 1, \forall v \in \mathcal{V}\right\}, \tag{12}$$

where In(v) denotes the set of incoming edges to vertex v.

CORNETO's generalized joint network inference problem from prior knowledge. We are now ready to introduce the general optimization problem that CORNETO proposes, which generalizes network inference methods from prior knowledge to a joint optimization with a common structure for all methods within the framework.

The objective of this joint optimization is to estimate the network structure for each sample by leveraging information from all samples, thereby improving accuracy and reducing variance in the predicted networks. CORNETO formulates this problem as a constrained optimization task, incorporating a structured sparsity-inducing penalty through network flows.

Given the union (annotated) graph $\mathcal{H}_u$, the general optimization problem is defined as

$$\begin{aligned} \min_{\mathbf{X},\mathbf{Y}} \quad & \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{X}_{:,i}, \mathbf{Y}_{:,i}; \mathcal{H}_u) + \lambda \parallel \mathbf{Y}\mathbf{1}_n \|_0 \\ \text{s.t.} \quad & \mathbf{X} \in \Omega_{\text{Flow}}(\mathcal{H}_u) \\ & \mathbf{Y} \in \Omega_{\text{Free}}(\mathbf{X}) \\ & (\mathbf{X}, \mathbf{Y}) \in \Omega_{\mathcal{M}}(\mathbf{X}, \mathbf{Y}), \end{aligned} \tag{13}$$

where

- $f(\mathbf{X}_{:,i}, \mathbf{Y}_{:,i}; \mathcal{H}_u)$ is the linear objective function term for sample i. The inclusion of $\mathcal{H}_u$ indicates that the specific values for the variables of the problem are derived from data corresponding to sample i within the annotations of the union graph $\mathcal{H}_u$. For example, if f involves edge costs for sample i, these costs are extracted from the ith component of the edge data vectors $\mathbf{d}_e$ (where $e \in \mathcal{E}_u$) that were established by the data mapping function ϕ.
- $\mathbf{X} \in \mathbb{R}^{|\mathcal{E}_u|\times n}$ represents the network flows variable, where each row corresponds to the flow of an edge and each column corresponding to the flows of a sample.
- $\mathbf{Y} \in \{0,1\}^{|\mathcal{E}_u|\times n}$ is a binary indicator matrix, such that $\mathbf{Y}_{i,j} = 0 \Rightarrow \mathbf{X}_{i,j} = 0$. This variable can be used to block flows through specific edges in the network.
- $\Omega_{\mathcal{M}}$ are method-specific constraints involving $\mathbf{X}$ and $\mathbf{Y}$.
- λ is a regularization parameter controlling sparsity across samples.
- $\mathbf{1}_n$ is a vector of ones of length n.

The structured sparsity-inducing penalty $\lambda\|\mathbf{Y}\mathbf{1}_n\|_0$, where λ is a regularization parameter controlling the trade-off between sparsity and fit, enables joint inference by minimizing the number of active edges (non-zero entries in $\mathbf{Y}$, which are edges potentially carrying flow) that are present in any sample or condition. The ℓ_0 regularization can be implemented in a MILP problem by linearizing the expression (Supplementary Information).

CORNETO's network inference method definition

Based on the previously defined components, a network inference method in this framework is defined as a tuple $\mathcal{M} = (\phi, \psi, \mathcal{P}_{\mathcal{M}})$, with $\mathcal{P}_{\mathcal{M}} = (f, \Omega_{\mathcal{M}})$.

Given a method $\mathcal{M}$, a PKN $\mathcal{H}$ and a data matrix $\mathbf{D} \in \mathbb{R}^{m\times n}$, the general optimization problem is constructed by applying the following steps:

(1) $\mathcal{H}' = \phi(\mathcal{H}, \mathbf{D})$
(2) $\mathcal{H}''_i = \psi(\mathcal{H}'), \quad \forall i \in \{1, 2, \ldots, n\}$
(3) $\mathcal{H}_u = \left(\bigcup_{i=1}^{n} \mathcal{V}''_i, \bigcup_{i=1}^{n} \mathcal{E}''_i\right)$
(4) Build $\mathcal{P}$ by plugging $\mathcal{P}_{\mathcal{M}}$ into the global problem template (equation (13)).

Shortest paths. We begin by illustrating how the simple shortest path problem can be implemented in this framework, and extended to multi-sample settings.

Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathbf{d} \in \mathbb{R}_+^{|\mathcal{E}|}$ representing the non-negative costs associated with the edges, and two vertices $s, t \in \mathcal{V}$, the shortest path problem can be reformulated as a flow optimization problem[73]. To do this, we need to transform the original graph into $\mathcal{G}'$ by adding two edges $(\varnothing, \{s\})$ and $(\{t\}, \varnothing)$ to inject or extract flow through the s–t vertices, and then we can use $\Omega_{\text{Flow}}(\mathcal{G})$, defined previously:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{d}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x} \in \Omega_{\text{Flow}}(\mathcal{G}) \\ & \mathbf{x}_{(\varnothing,s)} = 1 \\ & \mathbf{x}_{(t,\varnothing)} = 1. \end{aligned} \tag{14}$$

The objective is to minimize the total cost of flow, where $\mathbf{x}$ represents the flow along the edges, constrained by $\mathbf{x}_{\min} = 0$ and $\mathbf{x}_{\max} = 1$. The constraints $\mathbf{x}_{(\varnothing,s)} = 1$ and $\mathbf{x}_{(t,\varnothing)} = 1$ inject and extract one unit of flow at the source s and target t, respectively, and also guarantee that the solution is not the trivial zero flow. These constraints are specific to the shortest path method.

Multi-sample shortest paths. Given that we can reformulate the shortest path problem in CORNETO using flows, the method can be extended to handle multiple samples simultaneously, where each sample has its own source, target, and edge costs. In this case, instead of a single cost vector $\mathbf{c}$, we have a matrix $\mathbf{D} \in \mathbb{R}_{+}^{|\mathcal{E}|\times n}$, where each column corresponds to the costs for a different sample, as well as a list of source vertices $\{s_1, s_2, \ldots, s_n\}$ and a list of target vertices $\{t_1, t_2, \ldots, t_n\}$ for the n samples.

The goal is to solve the shortest path problem for all n samples simultaneously, while also introducing a regularization term to promote edge sparsity across samples. This is formulated as a joint optimization problem:

$$\begin{aligned}
\min_{\mathbf{X}} \ & \frac{1}{n}\sum_{i=1}^{n} \mathbf{D}_{:,i}^{\top}\mathbf{X}_{:,i} + \lambda \parallel \mathbf{Y}\mathbf{1}_n \parallel_0 \\
\text{s.t.} \ & \mathbf{X} \in \Omega_{\text{Flow}}(\mathcal{G}) \\
& \mathbf{Y} \in \Omega_{\text{Free}}(\mathbf{X}) \\
& \mathbf{X}_{i,j} = 1 \quad \forall i | e_i = (\varnothing, \{s_j\}), j = 1, \ldots, n \\
& \mathbf{X}_{i,j} = 0 \quad \forall i | e_i = (\varnothing, \{s_k\}), \forall k \neq j, j = 1, \ldots, n \\
& \mathbf{X}_{i,j} = 1 \quad \forall i | e_i = (\{t_j\}, \varnothing), j = 1, \ldots, n \\
& \mathbf{X}_{i,j} = 0 \quad \forall i | e_i = (\{t_k\}, \varnothing), \forall k \neq j, j = 1, \ldots, n,
\end{aligned} \tag{15}$$

where $s_1, s_2, \ldots, s_n$ are the input vertices for the shortest path associated with each sample, and $t_1, t_2, \ldots, t_n$ are the target or destination vertices.

The last four constraints involving the flow matrix $\mathbf{X}$ ensure that, for each sample j, flow is injected into the graph only through the edges connected to the specific source node s_j and is extracted only through the edges connected to the specific target node t_j. All other edges involving source or target vertices not associated with sample j have no flow. This guarantees that the flow is restricted to the relevant edges for each sample, maintaining the integrity of the source–target pair for the shortest path. It should be noted that, when there is one single sample, these constraints reduce to the ones in the single shortest path (equation (14)).

The single- and multi-sample shortest path problem can then be defined in CORNETO as $\mathcal{M}_{SP} = (\phi, \psi, \mathcal{P}_{SP})$ and $\mathcal{P}_{SP} = (f, \Omega_{SP})$, where

- $\phi(\mathbf{D}, \mathcal{G}) = (\{v' \mid v' = (v, \mathbf{0})\}, \{e' \mid e' = (e, \mathbf{d}_e)\})$, that is, data mapping trivially maps the data (weights or cost of the edges) to the edges of the graph, and vertices do not have any cost associated to them.
- $\psi(\mathcal{G}') = (\mathcal{V}', \mathcal{E}' \cup \{(e_s, \mathbf{0}), (e_t, \mathbf{0})\}$, where $e_s = (\varnothing, \{s\})$ and $e_t = (\{t\}, \varnothing)$, and $s, t \in \mathcal{V}'$ are the provided source and target vertices.
- $f(\mathbf{x}) = \mathbf{d}^{\top}\mathbf{x}$, where $\mathbf{d}$ is a column vector with the costs of the edges for a given sample, and $\mathbf{x}$ is the variable vector of flows (for a given sample) provided by CORNETO in the general problem (equation (13)).
- Ω_{SP} corresponds to the last four constraints of equation (15) involving the injection or extraction of a unit of flow, which is the method specific set of constraints to formulate the shortest path as a flow problem.

By plugging f and Ω_{SP} into equation (13), we already have multi-sample version of the shortest path problem in CORNETO. Note that, if there is only one sample and $\lambda = 0$, the problem is equivalent to a standard shortest path. If there is only one sample and $\lambda > 0$, then the single shortest path is regularized so, among all the paths with the mininum cost (if there is more than one solution), the shortest one (less number of edges) is returned. If there are n samples and $\lambda = 0$, then the problem is equivalent to solve independently each shortest path.

Supplementary Fig. 2 shows an example for all the components of the framework for the single shortest path problem.

Steiner trees. The Steiner tree problem can be defined as follows. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and a set of terminal vertices $T \subseteq \mathcal{V}$ with $|T| \geq 2$, the goal is to find a tree $S = (\mathcal{V}_S, \mathcal{E}_S)$ within the graph $\mathcal{G}$ that includes all the terminal vertices, that is, $T \subseteq \mathcal{V}_S$, while minimizing the total weight of the edges in S.

Steiner trees have two basic constraints: (1) all terminals must be selected as part of the solution; and (2) the solution is a tree. The reformulation of Steiner trees in CORNETO thus requires expressing them using the concepts defined in the framework. First, for the first constraint, the initial graph is transformed such that a new edge is added to every terminal vertex. One vertex is chosen as a root, and an edge to inject flow is added, and for the remaining terminals, we add one edge to extract flow. If no root is provided, the first terminal is selected as a root. It should be noted that, in a Steiner tree (undirected), any terminal vertex could be a root because all the terminals have to be connected. A constraint is added such that the input edge injects $u \geq 1$ units of flow, and every other edge has to extract at most $\frac{u}{|T|-1}$ of the injected flow. This constrains the space of feasible solutions to the space of subgraphs where the terminals are connected to the root through any path from the PKN.

The method SteinerTree, which uses a directed graph $\mathcal{G}$, can be implemented as follows:

- $\mathcal{G}' = \phi(\mathbf{D}, \mathcal{G}) = (\{v' \mid v' = (v, \mathbf{0})\}, \{e' \mid e' = (e, \mathbf{d}_e)\})$
- $\mathcal{G}'' = \psi(\mathcal{G}') = (\mathcal{V}', \mathcal{E}' \cup \{((\varnothing, r_j), \mathbf{0})\} \cup \{((t, \varnothing), \mathbf{0}) | t \in T \setminus \{r_j\}\}), \forall j \in \{1, 2, \ldots, n\}$ that is, an edge to inject flow is added to the root vertex, and one edge to extract flow is added to every other terminal that is not the root. These edges have no cost ($\mathbf{d}_e = \mathbf{0}$). There can be one different root and set for terminals per sample, indexed by j.
- Ω_{ST} further restricts the variables of the main problem, by imposing:
  - $\mathbf{Y} \in \Omega_{\text{Tree}}(\mathcal{G}_u)$, where $\mathcal{G}_u$ is the union graph of the n modified graphs $\mathcal{G}''$.
  - For each sample s, the flow injected into the root terminal r_j through the edge $(\varnothing, r_j)$ must be equal to $u \geq 1$:
  $$\mathbf{X}_{(\varnothing, r_j), j} = u \quad \forall j \in \{1, 2, \ldots, n\}.$$
  - For each sample s, the flow extracted from each non-root terminal $t \in T_j \setminus \{r_j\}$ through the edge $(t, \varnothing)$ must be equal to $\frac{u}{|T_j|-1}$:
  $$\mathbf{X}_{(t,\varnothing), j} = \frac{1}{|T_j| - 1} \quad \forall j, \forall t \in T_j \setminus \{r_j\}.$$
  - Each edge that injects flow through a root defined for a different sample cannot inject flow in the current sample:
  $$\mathbf{X}_{(\varnothing, r), j} = 0 \quad \forall j, \forall r \neq r_j.$$
  - Similarly, for the edges to extract flow through the terminals,
  $$\mathbf{X}_{(t,\varnothing), j} = 0 \quad \forall j, \forall t \notin T_j.$$
- $f(\mathbf{y}) = \mathbf{d}^{\top}\mathbf{y}$, which minimizes the selection of edges in the tree.

For directed Steiner trees, the root node must be provided, because the reachability of vertices changes depending on the directions of the edges. Directionality of flows is controlled by lower and upper bounds on the edges. If an edge has a negative lower bound and positive upper bound, the edge behaves as a undirected edge. If the lower bound is 0, the edge is directed.

PCST. The PCST problem can be defined as follows. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a non-negative prize function $p : V \to \mathbb{R}^+$ that assigns a prize to each vertex, along with a weight function $w : E \to \mathbb{R}^+$ that assigns a cost to each edge, the goal is to find a tree $S = (\mathcal{V}_S, \mathcal{E}_S)$ within the graph $\mathcal{G}$ that minimizes the total cost of the edges minus the total prize of the vertices included in the tree, minimizing

$$\sum_{e \in E_S} w(e) - \sum_{v \in V_S} p(v).$$

The solution to the PCST problem balances the cost of connecting the vertices (using the edges) against the benefits (prizes) obtained by including vertices in the tree.

PCST in CORNETO is implemented using the Steiner tree implementation, with minimal changes. First, because terminals are not forced to be selected but they are part of the optimization, flow is not forced through them. Instead, the selection of prized terminals is moved to the objective function. Thus, the objective function is defined as

$$f(\mathbf{y}) = \mathbf{d}^\top \mathbf{y} - \mathbf{p}^\top \mathbf{y},$$

where $\mathbf{d} \in \mathbb{R}^{|\mathcal{E}_u|}$ is the vector containing the edge costs for a given sample, and $\mathbf{p} \in \mathbb{R}^{|\mathcal{E}_u|}$ the vector with prizes (associated to the attached edges for removing flow through the prized terminals) taken from the annotated union graph $\mathcal{G}_u$.

It should be noted that the vectors $\mathbf{d}$ and $\mathbf{p}$ are mutually exclusive, that is, for any index i, if $d_i \neq 0$, then $p_i = 0$, and if $p_i \neq 0$, then $d_i = 0$. This condition ensures that an edge either contributes to the edge cost (through $\mathbf{d}$) or is associated with a prize (through $\mathbf{p}$), but not both, making the problem equivalent to the original definition of the PCST.

FBA. FBA[16] is a mathematical approach used to analyse the flow of metabolites through a metabolic network. Given a metabolic network represented by a stoichiometric matrix $\mathbf{S}$, where each element $\mathbf{S}_{i,j}$ indicates the stoichiometric coefficient of metabolite i in reaction j, FBA aims to find the distribution of fluxes $\mathbf{v} \in \mathbb{R}^n$ through the network that maximizes or minimizes a particular objective function, such as the biomass production of a given organism.

The key assumptions and components of FBA are as follows:

- Steady-state assumption: the system is assumed to be in a steady state, meaning that the concentration of each metabolite does not change over time. The flux represents the rate at which each reaction occurs, typically expressed in units of millimoles per gram of dry weight per hour.
- Constraints: the fluxes are subject to constraints based on reaction capacities.
- Objective function: a linear objective function of the fluxes, such as maximizing biomass yield (for example, output flux from an artificial biomass reaction in the metabolic network).

A genome-scale metabolic network can be modelled in CORNETO using a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, by setting the coefficients of the incidence matrix $\mathbf{A}$ to the stoichiometric coefficients of the metabolic network. Then, the stoichiometric matrix $\mathbf{S}$ is equivalent to the incidence matrix of the hypergraph. By using the lower and upper bounds on reaction fluxes, the set of feasible metabolic fluxes is exactly the set of feasible flows in the hypergraph $\Omega_{\text{Flow}}(\mathcal{H})$ (equation (5)). Thus, FBA is a particular instance of a single-sample (hyper)network flow on CORNETO with $\lambda = 0$.

The objective function is defined as

$$f(\mathbf{x}) = -\mathbf{c}^\top \mathbf{x},$$

where $\mathbf{c} = \{0, 1\}^{|\mathcal{E}_u|}$ is a provided constant vector indicating the reaction fluxes to be maximized.

The FBA method does not need data mapping or graph transformations, because no omics data are used, and the PKN typically contains import and export reactions such that the space of feasible solutions contain non-zero flux (flows) solutions. CORNETO provides a method that uses COBRApy[74] to import any genome-scale metabolic network as a CORNETO hypergraph.

sFBA. sFBA[31,75] is a variant of FBA that seeks to find an optimal flux distribution with the additional constraint of minimizing the number of active reactions (reactions carrying non-zero flux) in the metabolic network. The goal is to identify the sparsest set of reactions that can support a given metabolic objective (for example, maximizing biomass production). This method is particularly useful for identifying core metabolic functions or simplifying metabolic networks for analysis.

To find the smallest set of active reactions (that is, those with non-zero fluxes) in a metabolic network that maximizes a given linear objective function, sFBA can be exactly implemented by activating the ℓ_0 penalty on the selected edges $\mathbf{y}$ ($\lambda > 0$), and adding a constraint on the fluxes, for example, $\mathbf{x}_{\text{biomass}} \geq 0.95 \times \mathbf{x}^*_{\text{biomass}}$, where $\mathbf{x}^*_{\text{biomass}}$ is the optimal value of biomass flux in the standard FBA solution. This constraint ensures that the sparse solution retains at least 95% of the maximum possible biomass production, while minimizing the number of active reactions.

The multi-sample sFBA method implemented with CORNETO extends this definition to find a metabolic network per condition such that the union of the networks is minimal, while still satisfying all the metabolic constraints of each condition, such as different cell substrates, different reaction flux bounds or any other type of constraint typically used in FBA modelling. This is achieved by reformulating the sFBA problem using CORNETO's framework, which creates n simultaneous flow problems that are equivalent to n coupled FBA problems. The metabolic fluxes for each problem are captured in the $\mathbf{X} \in \mathbb{R}^{|\mathcal{E}_u| \times n}$ variable created by CORNETO, and the flux indicator variable $\mathbf{Y} \in \{0, 1\}^{|\mathcal{E}_u| \times n}$, which indicates if reaction i in condition j can carry metabolic flux or is blocked in all samples. Instead of minimizing the selection of reactions with non-zero flux within each individual condition, we aim to minimize it across all samples simultaneously. We define sparsity as the minimum number of different selected reactions. Thus, maximizing sparsity across samples is equivalent to minimizing the total number of unique reactions in the union of the n different inferred metabolic networks. This definition of structured sparsity is exactly what CORNETO captures with the regularization term $\lambda \|\mathbf{Y}\mathbf{1}_n\|_0$.

It should be noted that, in sFBA, the only term in the objective function is the minimization of active reactions. As a result, increasing λ does not have a progressive effect; it only distinguishes between $\lambda = 0$ (no sparsity) and $\lambda > 0$ (sparsity applied).

Multi-sample sFBA for flux adjustment. We extended the idea of sFBA to support flux adjustment by incorporating an additional ℓ_2 regularization on the flux values (with parameter β) to control their magnitude, while maintaining global sparsity through a structured ℓ_0 penalty (with parameter λ). The method minimizes the discrepancy between the predicted and observed fluxes, subject to the same metabolic constraints as standard sFBA. The objective functon that optimizes is

$$\min_{\mathbf{X},\mathbf{Y}} \| \mathbf{M} \odot (\mathbf{X} - \hat{\mathbf{X}}) \|_{2,1} + \beta \| \mathbf{X} \|_{2,1} + \lambda \| \mathbf{Y} \|_0$$

where $\|\cdot\|_F$, where $\mathbf{M} \in \{0, 1\}^{|\mathcal{E}| \times n}$ masks the unmeasured fluxes, and $\| \mathbf{X} \|_{2,1} := \sum_{i=1}^n \| \mathbf{X}_{:,i} \|_2$ denotes the column-wise mixed $\ell_{2,1}$ norm.

It is important to note that the use of ℓ_2 regularization introduces a quadratic term in the objective function, converting the problem into a mixed integer quadratic program (MIQP), also supported in CORNETO.

Metabolic network inference from omics (iMAT). The iMAT method[27] is a FBA-based method used to reconstruct context-specific metabolic networks by integrating omics data with a genome-scale metabolic model. The method identifies a subset of reactions that are consistent with the omics data while ensuring that the resulting network satisfies the constraints imposed by FBA.

Given a metabolic network, and an omics-based classification of reactions into three categories:

- $C_H \subseteq \{1,\ldots,|\mathcal{E}_u|\}$: high-confidence reactions, which are strongly supported by omics data.
- $C_L \subseteq \{1,\ldots,|\mathcal{E}_u|\}$: low-confidence reactions, which are weakly supported by omics data.
- $C_U = \{1,\ldots,|\mathcal{E}_u|\} \setminus (C_H \cup C_L)$.: uncertain reactions, with ambiguous or no supporting data.

The iMAT method solves the following optimization problem:

$$\max_{\mathbf{y}\in\{0,1\}^n} \sum_{j\in C_H} \mathbf{y}_j - \sum_{j\in C_L} \mathbf{y}_j.$$

Subject to the FBA constraints. The binary variable $y_j \in \{0,1\}$ ensures that a reaction j can carry flux only if it is included in the network (that is, $y_j = 1$).

In CORNETO, this method is implemented similarly to the sFBA problem, but using the non-zero indicator $\mathbf{Y} \in \Omega_{\neq 0}(\mathbf{X})$ instead of Ω_{Free}. Omics data are mapped using the gene–protein reaction rules in the metabolic network, with a specific $\phi(\mathbf{D},\mathcal{H})$ (Supplementary Information). The objective function is then

$$f(\mathbf{y}) = \mathbf{d}_h^{\top}(1-\mathbf{y}) + \mathbf{d}_l^{\top}\mathbf{y},$$

where $\mathbf{d}_h = \mathbf{1}_{C_H} \in \{0,1\}^n$ and $\mathbf{d}_l = \mathbf{1}_{C_L} \in \{0,1\}^n$, with $\mathbf{1}_S$ being the indicator function.

Sign-consistent intracellular signalling network (CARNIVAL). We implemented the CARNIVAL method using the flow-based formulation for multi-sample signalling network inference from transcriptomics. To compare with the original Carnival R for single samples, we implemented a more flexible and efficient version of Carnival R (Supplementary Information).

Given a directed (signed) PKN $\mathcal{G}$, where $\mathbf{s} \in \{-1,1\}^{|\mathcal{E}|}$ is the vector of edge interaction signs (−1, inhibition; 1, activation) from the PKN, a matrix $\mathbf{D}^{|\mathcal{V}|\times n}$ with TF activities (where negative values indicate inhibition and positive values indicate activation), and $\mathbf{P} \in \{-1,1\}^{|\mathcal{E}_u \times n|}$ is a matrix indicating which vertices from the PKN are perturbed in each sample.

- $\phi(\mathbf{D},\mathcal{G}) = (\{v' \mid v' = (v, \mathbf{d}_v)\}, \{e' \mid e' = (e, \mathbf{0})\})$.
- $\psi(\mathcal{G}') = (\mathcal{V}', \mathcal{E}' \cup \{(\varnothing,\{v_i\}) \mid \mathbf{p}_i \neq 0\} \cup \{(\{v_k\},\varnothing) \mid \mathbf{d}_k \neq 0\})$, where:
  - $\mathbf{p}_i$ represents the ith entry of a given column j of $\mathbf{P}$ (that is, $\mathbf{p}_i = \mathbf{P}_{i,j}$), which corresponds to the perturbation status of vertex v_i in sample j (with $\mathbf{p}_i \neq 0$ indicating that vertex v_i is perturbed in this sample),
  - $\mathbf{d}_k$ represents the kth entry of the same column j of $\mathbf{D}$ (that is, $\mathbf{d}_k = \mathbf{D}_{k,j}$), which corresponds to the TF activity of vertex v_k in sample j (with $\mathbf{d}_k \neq 0$ indicating that TF v_k is active or inhibited in this sample).
- Ω_{Carnival} has the following constraints and variables:

$$\begin{aligned}
&\mathbf{Y} \in \Omega_{\text{Tree}}(\mathbf{X})\\
&\mathbf{E}^{\text{act}} + \mathbf{E}^{\text{inh}} \leq \mathbf{Y}\\
&\mathbf{V}^{\text{act}} + \mathbf{V}^{\text{inh}} \leq \mathbf{1}\\
&\mathbf{E}^{\text{act}} \leq (\mathbf{H}^{\top}\mathbf{V}^{\text{act}}) \odot \mathbf{1}_{\mathbf{s}>0} + (\mathbf{H}^{\top}\mathbf{V}^{\text{inh}}) \odot \mathbf{1}_{\mathbf{s}<0}\\
&\mathbf{E}^{\text{inh}} \leq (\mathbf{H}^{\top}\mathbf{V}^{\text{act}}) \odot \mathbf{1}_{\mathbf{s}<0} + (\mathbf{H}^{\top}\mathbf{V}^{\text{inh}}) \odot \mathbf{1}_{\mathbf{s}>0}\\
&\mathbf{V}^{\text{act}}_{i,j} - \mathbf{V}^{\text{inh}}_{i,j} = \mathbf{P}_{i,j} \quad \text{if} \quad \mathbf{P}_{i,j} \neq 0,
\end{aligned}$$

where $\mathbf{H} \in \{0,1\}^{|\mathcal{V}_u|\times|\mathcal{E}_u|}$ is defined as

$$\mathbf{H} = \max(-\mathbf{A}, 0) = \begin{cases} 1 & \text{if vertex } v \text{ is the tail of edge } e,\\ 0 & \text{otherwise}. \end{cases}$$

- $f(\mathbf{v}) = (\mathbf{1} - \mathbf{v} \odot \operatorname{sign}(\mathbf{d}))^{\top}|\mathbf{d}|$, where $\mathbf{v} = \mathbf{V}^{\text{act}}_{:,j} - \mathbf{V}^{\text{inh}}_{:,j}$ and $\mathbf{d} = \mathbf{D}_{:,j}$, for a given sample j.

Data availability

All data generated during this study, along with the computational notebooks used for experiments and for downloading external datasets, are available via GitHub at https://github.com/saezlab/corneto-manuscript-experiments and at https://doi.org/10.24433/CO.6622821.v1. The yeast microarray dataset was originally generated by Kemmeren et al. (2014)[43] and is publicly available at https://deleteome.holstegelab.nl/data/downloads/deleteome_responsive_mutants_ex_wt_var_controls.txt. Transcriptomics and phosphoproteomics data from the lung adenocarcinoma cohort of the CPTAC were obtained using the CPTAC Python package at https://paynelab.github.io/cptac/. Preprocessed transcriptomics data for ponatinib-treated cancer cell lines were obtained from Paton et al. (2024)[76], originally generated as part of the PANACEA DREAM challenge[77]. The TCR dataset, originally used by Fortelny et al. (2020)[60], was downloaded from http://kpnn.computational-epigenetics.org. It was originally generated by Datlinger et al.[78] and is publicly available via Gene Expression Omnibus (GEO) under accession number GSE137554.

Code availability

The implementation of the framework is available via GitHub at https://github.com/saezlab/corneto under GNU General Public License v3.0. Notebooks with the code used to run the experiments in the Article are available via GitHub at https://github.com/saezlab/corneto-manuscript-experiments. A citable container with the code and data is available at https://doi.org/10.24433/CO.6622821.v1 (ref. 79). To facilitate long-term accessibility and maintenance, centralized access to the code, documentation and related resources is provided through the project's dedicated website at https://corneto.org.

References

1. Zitnik, M. et al. Current and future directions in network biology. *Bioinformatics Adv.* **4**, vbae099 (2024).
2. Barabasi, Albert-Laszlo & Oltvai, Z. N. Network biology: understanding the cell's functional organization. *Nat. Rev. Genet.* **5**, 101–113 (2004).
3. Alon, U. *An Introduction to Systems Biology: Design Principles of Biological Circuits* (Chapman and Hall/CRC, 2019).
4. Bonneau, R. Learning biological networks: from modules to dynamics. *Nat. Chem. Biol.* **4**, 658–664 (2008).
5. Le Novere, N. Quantitative and logic modelling of molecular and gene networks. *Nat. Rev. Genet.* **16**, 146–158 (2015).
6. Camacho, D. M., Collins, K. M., Powers, R. K., Costello, J. C. & Collins, J. J. Next-generation machine learning for biological networks. *Cell* **173**, 1581–1592 (2018).
7. Stuart, T. & Satija, R. Integrative single-cell analysis. *Nat. Rev. Genet.* **20**, 257–272 (2019).
8. Virshup, I. et al. The scverse project provides a computational ecosystem for single-cell omics data analysis. *Nat. Biotechnol.* **41**, 604–606 (2023).
9. Langfelder, P. & Horvath, S. Wgcna: an R package for weighted correlation network analysis. *BMC Bioinform.* **9**, 559 (2008).
10. Zitnik, M. et al. Machine learning for integrating data in biology and medicine: principles, practice, and opportunities. *Inf. Fusion* **50**, 71–91 (2019).
11. Lobentanzer, S., Rodriguez-Mier, P., Bauer, S. & Saez-Rodriguez, J. Molecular causality in the advent of foundation models. *Mol. Syst. Biol.* **20**, 848–858 (2024).

12. Ideker, T., Dutkowski, J. & Hood, L. Boosting signal-to-noise in complex biology: prior knowledge is power. *Cell* **144**, 860–863 (2011).
13. Bork, P. et al. Protein interaction networks from yeast to human. *Curr. Opin. Struct. Biol.* **14**, 292–299 (2004).
14. Garrido-Rodriguez, M., Zirngibl, K., Ivanova, O., Lobentanzer, S. & Saez-Rodriguez, J. Integrating knowledge and omics to decipher mechanisms via large-scale models of signaling networks. *Mol. Syst. Biol.* **18**, e11036 (2022).
15. Fang, X., Lloyd, C. J. & Palsson, B. O. Reconstructing organisms in silico: genome-scale models and their emerging applications. *Nat. Rev. Microbiol.* **18**, 731–743 (2020).
16. Orth, J. D., Thiele, I. & Palsson, B. Ø. What is flux balance analysis? *Nat. Biotechnol.* **28**, 245–248 (2010).
17. Ideker, T. & Krogan, N. J. Differential network biology. *Mol. Syst. Biol.* **8**, 565 (2012).
18. Gutenkunst, R. N. et al. Universally sloppy parameter sensitivities in systems biology models. *PLoS Comput. Biol.* **3**, e189 (2007).
19. Albert, I., Thakar, J., Li, S., Zhang, R. & Albert, R. Boolean network simulations for life scientists. *Source Code Biol. Med.* **3**, 16 (2008).
20. Abou-Jaoudé, W. et al. Logical modeling and dynamical analysis of cellular networks. *Front. Genet.* **7**, 94 (2016).
21. Fröhlich, F., Kaltenbacher, B., Theis, F. J. & Hasenauer, J. Scalable parameter estimation for genome-scale biochemical reaction networks. *PLoS Comput. Biol.* **13**, e1005331 (2017).
22. Bertsimas, D. & Weismantel, R. *Optimization Over Integers* (Dynamic Ideas, 2005).
23. Ford, D. R. & Fulkerson, D. R. *Flows in Networks* (Princeton Univ. Press, 2010).
24. Bertsekas, D. *Network Optimization: Continuous and Discrete Models* vol. 8 (Athena Scientific, 1998).
25. Lotfollahi, M. et al. Biologically informed deep learning to query gene programs in single-cell atlases. *Nat. Cell Biol.* **25**, 337–350 (2023).
26. Seninge, L., Anastopoulos, I., Ding, H. & Stuart, J. Vega is an interpretable generative model for inferring biological network activity in single-cell transcriptomics. *Nat. Commun.* **12**, 5684 (2021).
27. Shlomi, T., Cabili, M. N., Herrgård, M. J., Palsson, BernhardØ. & Ruppin, E. Network-based prediction of human tissue-specific metabolism. *Nat. Biotechnol.* **26**, 1003–1010 (2008).
28. Liu, A. et al. From expression footprints to causal pathways: contextualizing large signaling networks with CARNIVAL. *NPJ Syst. Biol. Appl.* **5**, 40 (2019).
29. Tuncbag, N. et al. Simultaneous reconstruction of multiple signaling pathways via the prize-collecting Steiner forest problem. *J. Comput. Biol.* **20**, 124–136 (2013).
30. Meléndez-Hevia, E. & Isidoro, A. The game of the pentose phosphate cycle. *J. Theor. Biol.* **117**, 251–263 (1985).
31. Fleming, RonanM. T. et al. Cardinality optimization in constraint-based modelling: application to human metabolism. *Bioinformatics* **39**, btad450 (2023).
32. Segre, D., Vitkup, D. & Church, G. M. Analysis of optimality in natural and perturbed metabolic networks. *Proc. Natl Acad. Sci. USA* **99**, 15112–15117 (2002).
33. Schellenberger, J. & Palsson, BernhardØ. Use of randomized sampling for analysis of metabolic networks. *J. Biol. Chem.* **284**, 5457–5461 (2009).
34. Rodríguez-Mier, P., Poupin, N., de Blasio, C., Le Cam, L. & Jourdan, F. DEXOM: diversity-based enumeration of optimal context-specific metabolic networks. *PLoS Comput. Biol.* **17**, e1008730 (2021).
35. Smith, A. C., Eyassu, F., Mazat, Jean-Pierre & Robinson, A. J. MitoCore: a curated constraint-based model for simulating human central metabolism. *BMC Syst. Biol.* **11**, 1–13 (2017).
36. Noor, E., Eden, E., Milo, R. & Alon, U. Central carbon metabolism as a minimal biochemical walk between precursors for biomass and energy. *Mol. Cell* **39**, 809–820 (2010).
37. Antonakoudis, A., Strain, B., Barbosa, R., Jimenez del Val, I. & Kontoravdi, C. Synergising stoichiometric modelling with artificial neural networks to predict antibody glycosylation patterns in Chinese hamster ovary cells. *Comput. Chem. Eng.* **154**, 107471 (2021).
38. Jiménez del Val, I. et al. CHOmpact: a reduced metabolic model of chinese hamster ovary cells with enhanced interpretability. *Biotechnol. Bioeng.* **120**, 2479–2493 (2023).
39. Robaina Estévez, Semidán & Nikoloski, Z. Context-specific metabolic model extraction based on regularized least squares optimization. *PLoS ONE* **10**, e0131875 (2015).
40. Vlassis, N., Pacheco, M. P. & Sauter, T. Fast reconstruction of compact context-specific metabolic network models. *PLoS Comput. Biol.* **10**, e1003424 (2014).
41. Becker, S. A. & Palsson, B. O. Context-specific metabolic networks are consistent with experiments. *PLoS Comput. Biol.* **4**, e1000082 (2008).
42. Robaina Estévez, Semidán & Nikoloski, Z. Generalized framework for context-specific metabolic model extraction methods. *Front. Plant Sci.* **5**, 491 (2014).
43. Kemmeren, P. et al. Large-scale genetic perturbations reveal regulatory networks and an abundance of gene-specific repressors. *Cell* **157**, 740–752 (2014).
44. Badia-i-Mompel, P. et al. decoupleR: ensemble of computational methods to infer biological activities from omics data. *Bioinformatics Adv.* **2**, vbac016 (2022).
45. Gillette, M. A. et al. Proteogenomic characterization reveals therapeutic vulnerabilities in lung adenocarcinoma. *Cell* **182**, 200–225 (2020).
46. Li, Y. et al. Proteogenomic data and resources for pan-cancer analysis. *Cancer Cell* **41**, 1397–1406 (2023).
47. Hegde, C., Indyk, P. & Schmidt, L. A nearly-linear time framework for graph-structured sparsity. In *International Conference on Machine Learning* 928–937 (PMLR, 2015).
48. Wu, B. & Ghanem, B. Lp-Box ADMM: a versatile framework for integer programming. *IEEE Trans. Pattern Anal. Mach. Intell.* **41**, 1695–1708 (2018).
49. Varma, A. & Palsson, B. O. Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type *Escherichia coli* w3110. *Appl. Environ. Microbiol.* **60**, 3724–3731 (1994).
50. Geris, L. & Gomez-Cabrero, D. *Uncertainty in Biology: A Computational Modeling Approach* (Springer, 2015).
51. Villaverde, A. F., Raimúndez, E., Hasenauer, J. & Banga, J. R. Assessment of prediction uncertainty quantification methods in systems biology. *IEEE/ACM Trans. Comput. Biol. Bioinformatics* **20**, 1725–1736 (2022).
52. Covert, M. W., Schilling, C. H. & Palsson, B. Regulation of gene expression in flux balance models of metabolism. *J. Theor. Biol.* **213**, 73–88 (2001).
53. Marmiesse, L., Peyraud, R. & Cottret, L. Flexflux: combining metabolic flux and regulatory network analyses. *BMC Syst. Biol.* **9**, 93 (2015).
54. Chandrasekaran, S. & Price, N. D. Probabilistic integrative modeling of genome-scale metabolic and regulatory networks in *Escherichia coli* and *Mycobacterium tuberculosis*. *Proc. Natl Acad. Sci. USA* **107**, 17845–17850 (2010).
55. Dugourd, A. et al. Causal integration of multi-omics data with prior knowledge to generate mechanistic hypotheses. *Mol. Syst. Biol.* **17**, e9730 (2021).
56. Dimitrov, D. et al. LIANA+ provides an all-in-one framework for cell–cell communication inference. *Nat. Cell Biol.* **26**, 1613–1622 (2024).

57. Heinken, A., Basile, A., Hertel, J., Thinnes, C. & Thiele, I. Genome-scale metabolic modeling of the human microbiome in the era of personalized medicine. *Annu. Rev. Microbiol.* **75**, 199–222 (2021).
58. Bunne, C. et al. How to build the virtual cell with artificial intelligence: priorities and opportunities. *Cell* **187**, 7045–7063 (2024).
59. Cui, H. et al. Towards multimodal foundation models in molecular cell biology. *Nature* **640**, 623–633 (2025).
60. Fortelny, N. & Bock, C. Knowledge-primed neural networks enable biologically interpretable deep learning on single-cell sequencing data. *Genome Biol.* **21**, 1–36 (2020).
61. Nilsson, A., Peters, J. M., Meimetis, N., Bryson, B. & Lauffenburger, D. A. Artificial neural networks enable genome-scale simulations of intracellular signaling. *Nat. Commun.* **13**, 3069 (2022).
62. Barsacchi, M., Terre, H. A. & Lió, P. GEESE: metabolically driven latent space learning for gene expression data. Preprint at *bioRxiv* https://doi.org/10.1101/365643 (2018).
63. Paton, V. et al. Networkcommons: bridging data, knowledge, and methods to build and evaluate context-specific biological networks. *Bioinformatics* **41**, btaf048 (2025).
64. Lawler, E. L. in *Annals of Discrete Mathematics* vol. 4, 251–263 (Elsevier, 1979).
65. Chandy, K. M. & Lo, T. The capacitated minimum spanning tree. *Networks* **3**, 173–181 (1973).
66. Mairal, J., Jenatton, R., Obozinski, G. & Bach, F. Convex and network flow optimization for structured sparsity. *J. Mach. Learn. Res.* **12**, 2681–2720 (2011).
67. Bertsimas, D. & Tsitsiklis, J. N. *Introduction to Linear Optimization* vol. 6 (Athena Scientific, 1997).
68. Klamt, S., Haus, Utz-Uwe & Theis, F. Hypergraphs and cellular networks. *PLoS Comput. Biol.* **5**, e1000385 (2009).
69. Bradley, S. P., Hax, A. C. & Magnanti, T. L. *Applied Mathematical Programming* (Addison-Wesley, 1977).
70. Hu, T. C. Multi-commodity network flows. *Operations Res.* **11**, 344–360 (1963).
71. Diamond, S. & Boyd, S. CVXPY: a Python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.* **17**, 1–5 (2016).
72. Sagnol, G. & Stahlberg, M. PICOS: a Python interface to conic optimization solvers. *J. Open Source Softw.* **7**, 3915 (2022).
73. Papadimitriou, C. H. & Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity* (Dover Publications, 1998).
74. Ebrahim, A., Lerman, J. A., Palsson, B. O. & Hyduke, D. R. COBRApy: constraints-based reconstruction and analysis for Python. *BMC Syst. Biol.* **7**, 74 (2013).
75. Lewis, N. E. et al. Omic data from evolved *E. coli* are consistent with computed optimal growth from genome-scale models. *Mol. Syst. Biol.* **6**, 390 (2010).
76. Paton, V. et al. Assessing the impact of transcriptomics data analysis pipelines on downstream functional enrichment results. *Nucleic Acids Res.* **52**, 8100–8111 (2024).
77. Douglass, E. F. et al. A community challenge for a pancancer drug mechanism of action inference from perturbational profile data. *Cell Rep. Med.* **3**, 100492 (2022).
78. Datlinger, P. et al. Pooled CRISPR screening with single-cell transcriptome readout. *Nat. Methods* **14**, 297–301 (2017).
79. Rodriguez-Mier, P., Garrido-Rodriguez, M., Gabor, A. & Saez-Rodriguez, J. CORNETO: unified knowledge-driven network inference from omics data. *CodeOcean* https://doi.org/10.24433/CO.6622821.v2 (2025).
80. Lu, H. et al. A consensus *S. cerevisiae* metabolic model Yeast8 and its ecosystem for comprehensively probing cellular metabolism. *Nat. Commun.* **10**, 3586 (2019).
81. Mitsos, A. et al. Identifying drug effects via pathway alterations using an integer linear programming optimization formulation on phosphoproteomic data. *PLoS Comput. Biol.* **5**, e1000591 (2009).
82. Türei, D. énes, Korcsmáros, Tamás & Saez-Rodriguez, J. OmniPath: guidelines and gateway for literature-curated signaling pathway resources. *Nat. Methods* **13**, 966–967 (2016).
83. Dittrich, M. T., Klau, G. W., Rosenwald, A., Dandekar, T. & Müller, T. Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics* **24**, i223–i231 (2008).
84. Hegde, C., Indyk, P. & Schmidt, L. A fast, adaptive variant of the Goemans–Williamson scheme for the prize-collecting Steiner tree problem. In *Workshop of the 11th DIMACS Implementation Challenge* https://people.csail.mit.edu/ludwigs/papers/dimacs14_fastpcst.pdf (2014).
85. Tuncbag, N. et al. Network-based interpretation of diverse high-throughput datasets through the omics integrator software package. *PLoS Comput. Biol.* **12**, e1004879 (2016).
86. Selby, D. A., Sprang, M., Ewald, J. & Vollmer, S. J. Beyond the black box with biologically informed neural networks. *Nat. Rev. Genet.* https://doi.org/10.1038/s41576-025-00826-1 (2025).

Acknowledgements

We thank C. Schmidt, M. Matabuena, D. Dimitrov, J. Carbonell, M. Kalxdorf, A. Nilsson, B. Dalpedri, S. Müller-Dott, I. Rigato, B. Wadie, F. Jourdan and N. Poupin for valuable input. We acknowledge funding from the European Union's Horizon 2020 Programme under grant agreement no. 951773 (PerMedCoE, https://permedcoe.eu/) and grant agreement no. 965193 (DECIDER, https://www.deciderproject.eu/) and the State Parliament of Baden-Württemberg for the Innovation Campus Health + Life Science Alliance Heidelberg Mannheim.

Author contributions

Conceptualization: P.R.-M. and J.S.-R. Methodology: P.R.-M. Software: P.R.-M., A.G. and M.G.-R. Formal analysis: P.R.-M., M.G.-R. and A.G. Investigation: P.R.-M. Visualization: P.R.-M, M.G.-R. and A.G. Validation: P.R.-M., M.G.-R. and A.G. Writing—original draft: P.R.-M. Writing—review and editing: P.R.-M, M.G.-R., A.G. and J.S.-R. Funding acquisition: J.S.-R. Supervision: J.S.-R.

Funding

Open access funding provided by European Molecular Biology Laboratory (EMBL).

Competing interests

J.-S.R. reports in the past 3 years funding from GSK and Pfizer and fees/honoraria from Travere Therapeutics, Stadapharm, Astex, Owkin, Pfizer, Grunenthal, Tempus AI and Moderna. A.G. receives fees and honoraria from Tempus AI and is currently an employee of Sanofi. The other authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-025-01069-9.

Correspondence and requests for materials should be addressed to Julio Saez-Rodriguez.

Peer review information *Nature Machine Intelligence* thanks David Gomez-Cabrero and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.