nature machine intelligence

Supplementary information

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

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

In the format provided by the authors and unedited

Supplementary Information

Supplementary Experiments

Optimal single and multi-sample Steiner tree methods for network inference

Since CORNETO can be used to model optimisation problems that obtain tree-like structures, we can use it to improve Steiner tree like methods used to explore Protein-Protein interaction (PPI) networks. Steiner trees [24] and their variations, such as Prize Collecting Steiner Trees (PCSTs) [1, 13], are optimisation methods commonly used to find the most efficient way of connecting selected vertices, or "terminals", within a network, aiming to minimise the overall connection cost. These methods are frequently applied in the context of protein-protein interaction (PPI) networks [5, 12, 30], facilitating the construction of networks that not only capture the measured or deregulated proteins, but also identify the minimal set of required interactions and additional (unmeasured) proteins from the prior knowledge network (PKN) to connect the measured proteins or biological entities of interest. However, given the computational complexity involved in solving Steiner tree problems, many existing methods rely on heuristic algorithms ([10, 14, 30]). These heuristics are designed to provide quick solutions by making educated guesses, which, while necessary to provide fast solutions, often result in sub-optimal trees with unknown performance guarantees. In contrast, our approach ensures that optimal solutions can be obtained with guarantees, and even if the search is stopped before reaching the optimal solution, we still have an upper bound on how far the obtained solution is from the true optimum. This provides a more reliable trade-off between computational efficiency and solution quality. Similar to the other methods, Steiner trees can also be extended to multi-sample settings, where we are not only interested in the optimal tree per sample but also in identifying optimal trees across samples simultaneously. We showcase the advantages of using the exact Steiner tree methods implemented with CORNETO on directed and undirected PPIs, for both single samples and multi-sample settings.

Optimal Steiner trees

We compare the exact Steiner tree implementation in CORNETO against the heuristic implementation included in NetworkX, a widely used network library for Python [10]. The cost of a Steiner tree is defined as the sum of the weights of all edges selected to form the tree connecting the set of terminal vertices. Given two Steiner trees, T_1 and T_2, T_1 is considered better than T_2 if: (1) it connects the same set of terminal vertices, and (2) its total edge cost is lower. If no weights are assigned, each edge is assumed to have a cost of 1, making the tree with fewer edges the better choice.

Solvers often spend a considerable amount of time proving optimality, even when the best solution is found early. To assess CORNETO under different time constraints, we tested with limits of 2 hours and 10 minutes. The 2-hour limit allows the solver to find and verify the optimal Steiner tree, while the 10-minute limit reflects a practical time frame for users requiring fast, optimal or near-optimal solutions without full optimality proof. A more detailed analysis of the computation time is provided in Suppl. Fig. S7.

We benchmarked the capabilities of the exact Steiner tree implementation in CORNETO to generate better Steiner trees both for single or multiple samples. For the single sample case, we generated random graphs using *barabasi_albert_graph* method from NetworkX with 2 edges per vertex, and with n.vertices $\in [250, 500, 750, 1000, 1250, 1500]$ and n.terminals $\in [10, 20, 30, 40, 50]$, generating in total $6 \times 5 = 30$ datasets. For each dataset, we compared the results from NetworkX against CORNETO.

On average, CORNETO obtains Steiner trees with a total edge cost that is 15.35% lower than those produced by the NetworkX heuristic, with the improvement growing with larger graphs (Fig. S1a). Even with the 10-minute limit, CORNETO remains close to optimal at 13.46%. While the NetworkX heuristic is extremely fast (under a second), CORNETO's time varies from 10 seconds up to the selected time limit, with substantial variability depending on the problem.

Multi-sample Prize Collecting Steiner Trees

Prize Collecting Steiner Trees (PCST) extend the idea of Steiner trees by allowing for a trade-off between collecting prizes (connecting terminal vertices) and adding edges to the tree. A PCST solution is considered better if it maximises the total collected prize while minimising the overall edge cost, striking an optimal balance between including relevant vertices and keeping the network concise.

To measure the improvement on the cost of PCSTs using the multi-sample version (Fig. S1b), we generated 3 datasets with 10 samples, and with different number of terminals and overlap of terminals across samples. The graph was generated using the NetworkX function *barabasi_albert_graph* with 1000 vertices and 3 edges per vertex. From this graph, a different number of terminals were selected by random, $n_{terminals} \in [5, 20, 50]$ and the number of common terminals across samples was $n_{common} \in [0, 5, 10]$. The prizes were assigned to each terminal by sampling a random value from a uniform distribution between 1 and 10, and edges were assigned a fixed cost of 0.01. The methods were run with $\lambda \in [1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0]$. By definition, the multi-sample version of PCST is, in the worst case, as good as the single sample PCST. As long as there are shared edges across samples, the multi PCST method can outperform the single sample version by reducing costs and improving connectivity.

When there are no common terminals across samples, the difference between the single and CORNETO's multi PCST is minimal, although the multi-sample is slightly better as there might exist alternative solutions with some common

edges by chance (Fig. S1b, leftmost plot). As we increase the number of terminals and shared terminals across samples, there is an increased probability of finding shared edges. At this point, CORNETO's multi-sample PCST finds solutions with significantly fewer unique edges across samples compared to the single PCST, as demonstrated by the increasing gap between the Pareto curves of the two methods.

It should be noted that, while the method is general and flexible, ensuring a meaningful trade-off between collected prizes and edge costs in the objective function requires appropriate scaling of these values. This step is application-specific, as the relative importance of edges and terminals may vary depending on the biological context and data type. CORNETO does not include a built-in strategy for this scaling, as it depends on user-defined modelling choices and the interpretation of input data. As such, scaling must be handled as a preprocessing step, guided by domain knowledge.

In addition to the synthetic benchmark, we illustrate with different examples how to use CORNETO's implementation of Steiner trees and PCSTs across various omics data analysis applications both in single and multi-sample scenarios.

Example 1: Joint inference of cell-type specific cytokine networks using multi-sample undirected Steiner trees

In this example, we aimed to infer cell-type specific molecular interaction networks using single-cell transcriptomics data. We began by retrieving data from the Cytokine Atlas [4] as preprocessed in Paton et al. 2024 [21], which profiles the responses of 17 immune cell types to 86 cytokines using single-cell transcriptomics. Since this analysis is based on mouse data, we imported a mouse-specific prior knowledge network. For this, we retrieved all interactions available in Omnipath for mouse, including gene-regulation interactions, protein-protein interactions, and signalling interactions, providing a comprehensive network to be used in subsequent analyses.

Next, we generated an overview of the differential data by discretising the gene expression results for each contrast as up- or down-regulated. Specifically, genes were labelled as "Up" if their log fold change (logFC) was greater than 1 and the adjusted p-value (padj) was less than 0.05. Similarly, genes were labelled as "Down" if the logFC was less than 0 and the padj was below the same threshold. Genes that did not meet these criteria were labelled as "Not significant".

For each cell type, we identified the 10 treatments with the largest number of differentially expressed genes. For each of these treatments, the deregulated genes were used as terminal vertices in the construction of a multi-sample Steiner tree, which captures the most relevant signalling pathways affected by the perturbations (figure S1c).

For each cell type, we used the differential response to cytokines as individual samples, without combining data from different cell types. These subnetworks, unique to each cell type, integrate genes responding to multiple perturbations and highlight key central proteins such as JUN and STAT3, both of which play crucial roles in immune responses.

This example demonstrates how CORNETO's multi-sample modelling can be leveraged to construct cell-type specific molecular networks from single-cell transcriptomics data.

Example 2: Joint network inference from proteomics data using multi-sample undirected PCSTs

Another interesting application of the multi-sample approach lies in patient cohort-level analysis. In the context of analysing omics data from patients with a specific disease, these methods help reduce inter-patient variability, enabling the identification of regulatory mechanisms and disease-associated biomarkers that may otherwise be masked by individual differences. For instance, an inferred node may connect to multiple proteins that are deregulated across several patients independently but not consistently enough across the cohort to register in a differential analysis. Our multi-sample analysis enables the identification of this type of patterns.

To illustrate this concept, we applied a multi-sample PCST analysis using data from a cluster of patients with similar proteomic profiles from the Clinical Proteomic Tumour Analysis Consortium (CPTAC) lung adenocarcinoma (LUAD) cohort. We focused on patients showing similar profiles between normal and tumour samples, assuming that a collective analysis could uncover consistent biological alterations and potential biomarkers.

To do this, we first computed differential proteomic profiles between tumour and normal samples from the CPTAC dataset. The Python package *cptac* was used to obtain the data. To ensure the analysis was relevant, we filtered the dataset to include only patients who had at least one tumour sample and one normal sample. This subset of patients, who meet the criteria of having at least two observations (including both tumour and normal samples), was selected for further analysis.

For prior knowledge, we extracted a generic protein-protein interaction (PPI) network from Omnipath (https://omnipathdb.org/interactions?genesymbols=1, accessed on September 2024), containing 81,528 interactions. To preprocess this network, we identified a set of proteins of interest by processing the proteomics data to focus on proteins that exhibit meaningful differences between normal and tumour samples across patients. First, we filtered out proteins not measured in all samples, ensuring consistency across the dataset. For each unique patient, we computed the log fold change in protein expression between tumour and normal samples. This results in a dataset where the log fold change for each protein is recorded for each patient.

Next, the protein differences were aggregated, and the average log fold change for each protein was calculated. To focus on proteins that were significantly altered, we applied a filter based on an absolute log fold change greater than 0.5. The proteins passing this threshold were considered the proteins of interest. These proteins were then used to filter the PPI network, retaining only the nodes corresponding to the significantly altered proteins. This ensures that the network represents the most relevant interactions in the context of the differential proteomics data. The largest connected

component of this filtered network was extracted for further analysis, forming a context-specific network from which we plan to extract Steiner trees to explore key biological pathways.

To illustrate how the multi-sample PCST method can be used in this type of settings, we applied Principal Component Analysis (PCA) to reduce the dimensionality of the patient data by transforming the scaled protein difference data into a smaller set of principal components (PCs). Four principal components were selected, capturing the most significant variance within the dataset. After the PCA transformation, we computed a pairwise correlation matrix of the patients. To focus on a specific set of patients, we identified the patient with the highest variability and selected the nine patients most correlated with this patient. This provided a targeted cluster of patients exhibiting the most similar patterns in their principal component profiles, helping to identify key clusters of patients with shared biological or clinical characteristics.

We then identified the proteins with the highest variability among the selected patients by calculating the variance in protein expression across this group. The proteins were ranked by their variability, and to focus on the most significant ones, we applied a quantile cut-off, selecting the proteins with variance above the 90th percentile. This allowed us to highlight the most variable proteins, which could be biologically relevant for understanding differences within this cluster of patients with similar profiles. We used the absolute value of these proteins as prizes for the PCST method.

The resulting optimal PCST network revealed several predicted Steiner vertices with well-established roles in cancer, including *CDK1* and *VCAN* (Fig. S1d, left). *CDK1* is recognized as a poor prognostic marker in lung adenocarcinoma [25]. Additionally, high expression levels of *VCAN* have been linked to significantly worse survival rates in LUAD [32]. To further explore the significance of selected vertices, we analysed their mutational burden, defined as the number of mutations present in one or more genes. Assuming that genes within the selected subnetworks are more relevant to disease progression, we compared the mutational burden in three networks: the context-free OmniPath PPI, the initial contextualised PKN, and the final context-specific PCST. Specifically, we measured the proportion of genes within each network that showed a number of missense mutations across all patients above a specific threshold. The optimal PCST showed a higher proportion of genes with 10 or more mutations when compared to the other two networks (Fig. S1d, right), likely due to the exclusion of sample-specific nodes and the prioritisation of genes more deregulated across all samples.

Example 3: Analysis of phosphoproteomics data with single condition directed PCSTs

In the context of cellular signalling, phosphoproteomics enables the measurement of thousands of phosphopeptides in a single experiment. These data can be used to infer signalling pathways through approaches such as PHONEMeS [9, 29], a method that contextualises kinase/phosphatase–substrate interaction networks. PHONEMeS aims to identify the smallest subnetwork that connects a known perturbation (e.g. a drug target) to as many deregulated phosphosites as possible. It effectively solves a Prize Collecting Steiner Tree (PCST) problem with directional constraints, ensuring signal propagation downstream from the perturbation source.

We extended PHONEMeS by implementing a more flexible version in CORNETO, reusing the multi-sample PCST method. As a case study, we re-analysed phosphoproteomics data from CPTAC-CCRCC, using as input the results of comparing tumour and adjacent normal tissue samples. Using OmniPath kinase–substrate interactions as a Prior Knowledge Network (PKN), we focused on the epidermal growth factor receptor (*EGFR*)—a common mutation target in renal carcinoma—as the source node for signalling propagation. The phosphoproteomic profiles from CPTAC-CCRCC were processed using FragPipeAnalyst [33], a state-of-the-art proteomics analysis suite. These data include log fold change (logFC) and adjusted P-values from the differential analysis.

We first loaded the phosphoproteomics dataset, which contains differential phosphorylation events between tumour and normal samples [21]. Next, we retrieved a prior knowledge network from the OmniPath [31] database, focusing on kinase–substrate interactions from the enzyme–PTM dataset. This dataset includes both phosphorylation and dephosphorylation events, which were filtered accordingly. Interactions were formatted to identify phosphosites as targets, and integrator nodes were added to connect phosphosites back to their corresponding proteins, enabling network flow through these regulatory interactions.

We defined *EGFR* as the root vertex, serving as the origin of signal propagation within the network. Phosphosites were categorised based on their differential regulation:

- **Deregulated:** adjusted P-value ≤ 0.01 and logFC ≥ 1,
- **Undetermined:** adjusted P-value ≤ 0.01 and logFC ≤ 1,
- **Not changing:** adjusted P-value ≥ 0.01.

The dataset comprises 34,048 phosphosites, of which 3,185 (9.35%) are reachable within the prior knowledge network. Among these, 2,337 were not changing, 696 were undetermined, and 152 were deregulated.

With the root vertex, prior knowledge network, and categorised phosphosites defined, we applied the directed PCST method. This approach extracts a subnetwork connecting the root to deregulated phosphosites while avoiding unchanged ones. Unchanged phosphosites were included only when essential to connect more than two deregulated targets.

The resulting network revealed increased activity in cancer-associated proteins such as CDK2, MAPK1, and GSK3β. Notably, GSK3β—a protein with a well-established role in renal carcinoma [2]—emerged as a key node, linking deregulated phosphosites downstream of SGK1 and CDK2 (Fig. S1e). This example demonstrates how our framework can reimplement and extend existing methods using the unified optimisation-based formulations provided by CORNETO.

Biologically-informed neural network for prediction of T-cell receptor activation from single cell RNA-seq

We implemented a version of the Knowledge-Primed Neural Network (KPNN), a type of artificial neural network whose architecture is constrained by prior biological knowledge to enhance interpretability and learning performance [7]. KPNNs require a PKN that specifies the possible biological interactions used to structure the neural network. We improve upon the original KPNN method by using CORNETO as an optimisation engine to automatically identify the minimal Directed Acyclic Graph (DAG) from a given PKN that connects T-cell receptor (TCR) stimulation to downstream genes. CORNETO employs its *Acyclic Flow* component to search the space of DAGs and minimise the number of selected edges, thereby identifying the smallest sub-network that retains full connectivity from the *TCR* receptor to the input genes.

After identifying the optimal DAG, CORNETO constructs the corresponding neural network architecture by performing a topological sort of the DAG vertices using Kahn's algorithm [15]. The neural network is then instantiated using the multi-backend Keras 3 library (https://keras.io), assigning a Dense layer with a single neuron per vertex. Each node also includes a batch normalisation layer (without learnable parameters), and dropout layers (by default, dropout = 0.20) are added to vertices with more than one upstream connection. Weights and biases are regularised using Elastic Net regularisation ($\ell_1 = 1\text{e}{-3}$, $\ell_2 = 1\text{e}{-2}$).

To test the method, we used the same CROP-seq dataset to predict TCR activation in Jurkat cells, along with the original PKN downloaded from http://kpnn.computational-epigenetics.org. We first built the KPNN using CORNETO and the original (non-optimised) PKN, resulting in a model with 26,236 trainable parameters. Performance was evaluated using a stratified 5-fold cross-validation strategy [22], yielding an average ROC AUC of 0.988. Subsequently, we used CORNETO to compress the PKN by identifying the smallest DAG that connects the input genes to the *TCR* receptor vertex. This yielded a reduced model with 12,459 parameters, corresponding to a 2.10× compression ratio. The compressed network retained the same average ROC AUC of 0.988, with no degradation in performance.

Supplementary Methods

Solver settings

For optimisation, we used the GUROBI solver (version 12) under an academic licence (https://www.gurobi.com/academia/academic-program-and-licenses/), and CVXPY as the optimisation backend (version 1.6). To improve solution precision for the MILP problems, we set the *IntegralityFocus* parameter to 1 for GUROBI. In some of the experiments, we also applied GUROBI's *NoRelHeurTime* option, which enabled the no-relaxation heuristic to make convergence faster.

Multi-sample CARNIVAL for intracellular signalling

To evaluate the performance of the multi-sample CARNIVAL method under controlled conditions, we used transcriptomic data from drug-treated cell lines provided by the PANACEA consortium [6]. From this dataset, we generated synthetic ground-truth signalling networks. Specifically, we selected samples treated with Ponatinib and estimated TF activities from differential expression between treated and untreated cells, using Decoupler. The known Ponatinib drug targets were obtained from [21] and used as common inputs across all samples. Seven distinct cell lines treated with Ponatinib were selected (H1793, LNCAP, KRJ1, HCC1143, EFO21, PANC1 and HF2597), resulting in seven samples with identical inputs but differing TF activities across samples. Using this data, we then ran CARNIVAL multi-sample with $\lambda \in \{0.0, 0.1, 0.2\}$ to generate three datasets (D1, D2, and D3, respectively). Solutions generated with larger λs are expected to have a higher degree of similarity.

Using these ground truth datasets, which reflect varying degrees of similarity across samples depending on the regularisation parameter λ, we evaluated the performance of CORNETO's multi-sample CARNIVAL method. To do so, we applied five-fold cross-validation by systematically holding out subsets of transcription factors—those originally used to generate the datasets—across all seven samples. The known Ponatinib drug targets were retained as common inputs, and only the remaining TFs were used during inference. This setup allowed us to assess whether incorporating structured sparsity via the λ parameter improves the recovery of the original signalling networks, particularly in cases where those networks were generated with higher λ values and thus exhibit greater inter-sample similarity. It should be noted that CARNIVAL cannot TFs that are not provided to the method as output vertices. As a result, standard classification metrics based on individual TF recovery cannot be applied directly. Instead, we evaluate performance by comparing the inferred signalling networks with the original ground truth networks using precision and recall. These metrics quantify how similar the reconstructed networks are when only a subset of TFs is used during optimisation, thereby assessing the method's ability to recover the underlying network structure under incomplete output information (Supplementary Fig. S6).

For the validation with real patient data, we used matched transcriptomic and phosphoproteomic profiles from lung adenocarcinoma (LUAD) patients in the CPTAC cohort [8, 17]. The dataset was accessed using the `cptac` Python package (https://github.com/PayneLab/cptac). To enrich for patients with deregulated signalling, we selected the top 10% of patients with the highest number of missense mutations, nonsense mutations, and frameshift indels. Differential transcriptomic and phosphoproteomic profiles were computed using paired tumour and adjacent normal tissue. TF activities were inferred using `decoupler`, and kinase activities were estimated from phosphoproteomic data using univariate linear

models. We used the OmniPath signalling network [11] as the PKN and treated all kinases in the network as candidate input nodes. We used the multi-sample method to infer networks under varying regularisation strengths λ, both in single- and multi-sample settings. In the single sample setting, the λ is equivalent to the original β regularisation in CARNIVAL which penalises the number of selected edges in the solution.

Validation was based on identifying shared deregulated kinases, defined as those with top-quartile absolute activity scores in at least 50% of patients. We assessed the total and intersecting network edges across patients, and measured the proportion of those edges linked to the validation kinase set. Each condition was repeated across multiple random seeds to evaluate robustness.

Sparse FBA simulations using MitoCore

The MitoCore metabolic network [27] was imported using the numpy-based compressed Genome-Scale Model from the MIOM package [23]. We generated data by simulating single knockout (KO) interventions in the metabolic network model. For each KO, we recorded two key outputs: the optimal flux representing biomass production (x^*_{ko}) and the number of active reactions (n_{ko}).

To select knockouts that caused significant changes in the network, we applied the following criteria:

- The difference in the number of active reactions between the knockout and wild-type ($|n_{\text{ko}} - n_{\text{wt}}|$) was greater than 10.
- The knockout resulted in non-zero biomass production, exceeding a minimal threshold ($x^*_{\text{ko}} > 10^{-3}$).
- The biomass production in the knockout was lower than that of the wild-type ($x^*_{\text{ko}} \leq 0.999 \cdot x^*_{\text{wt}}$).

These criteria produced a pool of 47 targetable reactions. From this pool, we generated datasets of varying sizes (2, 4, 8, or 16 KOs) by randomly selecting reactions to form each dataset. This process was repeated 10 times for each sample size, resulting in 10 datasets per size.

Next, we applied both the original sparse FBA and the multi-sample sparse FBA to each dataset. To investigate whether allowing metabolic flexibility in the KOs increases the likelihood of identifying shared metabolic pathways, we applied different constraints on the minimum biomass production for each KO. Specifically, we imposed as a constraint, to achieve at least 90% of the optimal biomass production for the KO, 50% and a more relaxed constraint of 10%.

We compared the resulting sparse networks obtained from applying sparse FBA on individual samples in each dataset with those derived from the CORNETO's multi-sample sparse FBA. Since the single-sample sparse FBA is applied separately to each sample, we compared the networks by taking the union of the sparse networks from the individual samples and comparing it to the union of the networks from the multi-sample sparse FBA. It is important to note that using the union of networks ensures a valid metabolic network, as taking the intersection would result in disconnected reactions that are common across samples, which would not represent a functional, simulable network.

The proportion of reactions selected by each method (single-sample and multi-sample sparse FBA) was calculated by first gathering the reactions selected from each dataset for both methods. These selected reactions were then combined across all datasets. For each reaction, the average frequency of selection was computed, which represents the proportion of times that reaction was chosen by each method. For example, a value of 1.0 for a reaction means that across the 10 replicates for each sample size, the reaction was active in all samples.

Flux adjustment simulation

To validate the sparse FBA method for flux adjustment, we sampled steady-state fluxes from the MitoCore v1.0.1 metabolic network using the OPTGP sampler [19] available in the COBRApy package, from which we extracted 10 random samples. We then performed cross-validation by withholding a varying percentages of observed fluxes across all samples and adding Gaussian noise to the held-out values. We measured the performance of the method by varying λ, and using $\beta = 1e-3$ for the ℓ_2 regularisation.

For the evaluation, we tested different settings, leaving out between 20% and 90% of the flux measurements, and adding different levels of Gaussian noise from $\sigma = 0.0$ (noiseless setting) to $\sigma = 5.0$

Preprocessing of the yeast transcriptomics data

We obtained the microarray data of controls and mutant yeast strains from [16] and the yeast metabolic network from [28] (version 8.5.0). To select the single KO yeast mutant samples for analysis, we calculated the proportion of genes showing significant changes compared to the control. We excluded the knocked-out genes and computed the 5th to 95th percentile range of the M/A ratio for all non-knocked-out genes across all mutants (using all genes, not only metabolic ones). Genes were considered to have changed significantly if their M/A ratio fell outside this percentile range (i.e., either above the 95th percentile or below the 5th percentile) and had a p-value ≤ 0.01. Mutants with more than 15% of metabolic-related genes (genes in the Gene-Protein-Reaction rules in the metabolic network) showing significant changes were selected for further analysis. This resulted in the selection of 12 samples (Table 1).

In order to infer metabolic networks from gene expression, we applied the standard pre-processing strategy used by iMAT [26]. We first classified genes into two sets: highly expressed and lowly expressed genes, using two quantile-based thresholds applied on the differential expression values (KO vs WT) of the genes per condition. Then, we assigned reaction scores by applying the Gene-Protein-Reaction (GPR) rules for the reactions (Suppl. Sec. "Implementation of ϕ

Mutant	Prop. of altered genes
opi3-del vs. wt	0.499091
ram1-del vs. wt	0.400909
cho2-del vs. wt	0.325455
aco1-del vs. wt	0.322727
pfk2-del vs. wt	0.210000
ipk1-del vs. wt	0.203636
vps15-del vs. wt	0.201818
sis2-del vs. wt	0.196364
arg82-del vs. wt	0.189091
plc1-del vs. wt	0.186364
rnr4-del vs. wt	0.180000
pfk27-del vs. wt	0.150909

Table 1 Table: Selected knockout (KO) yeast mutants with more than 15% of metabolic-related genes showing significant changes (M/A ratio outside the 5th to 95th percentile and p-value ≤ 0.01).

mapping function for Gene-Protein-Reaction rules in metabolic networks") in the yeast genome scale metabolic network to obtain the reaction scores. It should be noted that even though we used discrete reaction scores, as in the original iMAT, CORNETO's implementation supports also continuous scores, in a way that different reactions can have different importances in the optimisation.

For the discretisation, we applied the same threshold to assign a -1 to the gene if the M/A ratio was below 5th percentile, +1 if above 95th percentile, and 0 otherwise. Then, we applied the Gene-Protein-Reaction rules to obtain the reaction scores but using the discretised values instead of the continuous values.

For evaluation, we performed a 5-fold cross validation, leaving out a subset of the reaction scores for validation (across samples) for which we have assigned a -1 or +1. For each fold, a different subset of reactions for each sample is left out, and the remaining data is used for jointly inferring the metabolic networks. We expect that reactions with negative scores are not selected (0), and the ones with positive scores are selected (1), using only the training data. The binary vector of the predictions, extracted from the inferred networks, is compared with the validation data, after converting the -1 (negative score) to 0.

Linearisation of the ℓ_0 sparsity term

The multiplication of $\mathbf{Y}$ with the vector $\mathbf{1}_n$ in the structured sparsity-inducing penalty results in a vector $\mathbf{s} = \mathbf{Y}\mathbf{1}_n$ of size $|\mathcal{E}_u|$, where each element s_i represents the number of samples in which the corresponding edge i is active. The ℓ_0 can be implemented by linearising the expression through the introduction of a new binary variable $\mathbf{z} \in \{0,1\}^{|\mathcal{E}_u|}$.

Each element z_i serves as an indicator of whether edge i is active in at least one sample. Specifically, $z_i = 1$ if edge i is active, and $z_i = 0$ otherwise. To enforce this condition, we introduce the following constraint:

$$z_i \geq \frac{s_i}{n} \quad \forall i \in \{1, \ldots, |\mathcal{E}_u|\}$$

Here, s_i represents the number of samples in which edge i is active, and n is the total number of samples. This constraint ensures that $z_i = 1$ whenever edge i is active in at least one sample, as the normalised sum $\frac{s_i}{n}$ will be positive if any entry in the i-th row of $\mathbf{Y}$ (the binary indicator matrix for non-blocked edges) is non-zero. Conversely, if no sample uses edge i (i.e., $s_i = 0$), then $\frac{s_i}{n} = 0$, allowing $z_i = 0$.

Finally, the structured sparsity-inducing penalty term in the objective function is given by:

$$\lambda \sum_{i=1}^{|\mathcal{E}_u|} z_i$$

This penalty encourages sparsity by minimising the number of edges from the prior knowledge network (PKN) that can be selected across the samples. The binary nature of z_i ensures that each edge is only penalised once, regardless of how many samples use it.

CARNIVAL reimplementation

For completeness, we ported the original CARNIVAL R method [18, 20] with CORNETO, but without using the novelties proposed in this framework. The purpose is twofold: 1) To provide users with the old implementation in Python, enabling them to leverage the new API for modelling and additional solver support. 2) To allow a fair comparison between single and multi-sample approaches.

Given a directed signed PKN $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each edge has an interaction type interaction$(e) \in \{1, -1\}$, representing activation or inhibition.

Variables:

For each vertex $v \in V$:

- $V^{\text{act}}(v) \in \{0, 1\}$: Activation indicator.
- $V_{\text{inh}}(v) \in \{0, 1\}$: Inhibition indicator.
- $V_{\text{pos}}(v) \in [0, M]$: Positional value for acyclicity constraints.

For each edge $e \in E$:

- $E_{\text{act}}(e) \in \{0, 1\}$: Edge activation indicator.
- $E_{\text{inh}}(e) \in \{0, 1\}$: Edge inhibition indicator.

Parameters:

- Perturbations: A mapping $V_p \subseteq V \to \{\pm 1\}$.
- Measurements: A mapping $V_m \subseteq V \to \mathbb{R}$.
- $\beta \geq 0$: Regularisation weight.
- $M \geq 0$: Maximum positional value (default $M = |V|$).
- penalize $\in \{\text{nodes}, \text{edges}, \text{both}\}$: Regularisation type.

Objective Function:

$$\min_{V_{act}, V_{inh}} \sum_{v \in \mathcal{V}} |d_v| \left[1 - \sigma_v \left(V_{\text{act}}(v) - V_{\text{inh}}(v)\right)\right] + \beta \cdot \mathcal{R},$$

where:

- d_v is the measurement or perturbation value for vertex v.
- $\sigma_v = \text{sign}(d_v)$.
- $\mathcal{R}$ is the regularisation term defined as:

$$\mathcal{R} = \begin{cases} \mathcal{R}_V = \sum_{v \in V} \left(V_{\text{act}}(v) + V_{\text{inh}}(v)\right), & \text{if penalize} = \text{nodes}, \\ \mathcal{R}_E = \sum_{e \in E} \left(E_{\text{act}}(e) + E_{\text{inh}}(e)\right), & \text{if penalize} = \text{edges}, \\ \mathcal{R}_V + \mathcal{R}_E, & \text{if penalize} = \text{both}. \end{cases}$$

Constraints:

1. **Vertex activation/inhibition exclusivity:**

$$V_{\text{act}}(v) + V_{\text{inh}}(v) \leq 1, \quad \forall v \in V.$$

2. **Edge activation/inhibition exclusivity:**

$$E_{\text{act}}(e) + E_{\text{inh}}(e) \leq 1, \quad \forall e \in E.$$

3. **Sign consistency:**
   For each edge $e \in E$ from s_e to t_e:
   - If interaction$(e) = 1$ (activation):

$$E_{\text{act}}(e) \leq V_{\text{act}}(s_e), \quad E_{\text{inh}}(e) \leq V_{\text{inh}}(s_e).$$

   - If interaction$(e) = -1$ (inhibition):

$$E_{\text{act}}(e) \leq V_{\text{inh}}(s_e), \quad E_{\text{inh}}(e) \leq V_{\text{act}}(s_e).$$

4. **Acyclicity constraints**:

$$V_{\text{pos}}(t_e) - V_{\text{pos}}(s_e) \geq 1 - M \left[1 - \left(E_{\text{act}}(e) + E_{\text{inh}}(e)\right)\right], \ \forall e \in E.$$

5. **Signal propagation as a tree:**

$$\sum_{e \in \text{In}(v)} \left(E_{\text{act}}(e) + E_{\text{inh}}(e)\right) \leq 1, \quad \forall v \in V.$$

   where $\text{In}(v)$ is the set of edges ending at vertex v.
6. **Vertex activation/inhibition conditions:**

For each non-perturbed vertex $v \in V \setminus V_p$:

$$V_{\text{act}}(v) \leq \sum_{e \in \text{In}(v)} E_{\text{act}}(e), \quad V_{\text{inh}}(v) \leq \sum_{e \in \text{In}(v)} E_{\text{inh}}(e).$$

7. **Perturbed inputs:**
   For each perturbed vertex v provided as input:

- If activation (perturbation value = +1):

$$V_{\text{act}}(v) = 1, \quad V_{\text{inh}}(v) = 0.$$

- If inhibition (perturbation value = -1):

$$V_{\text{act}}(v) = 0, \quad V_{\text{inh}}(v) = 1.$$

Variable Domains:

- $V_{\text{act}}(v),\ V_{\text{inh}}(v) \in \{0, 1\},\ \forall v \in V.$
- $E_{\text{act}}(e),\ E_{\text{inh}}(e) \in \{0, 1\},\ \forall e \in E.$
- $V_{\text{pos}}(v) \in [0, M],\ \forall v \in V.$

Implementation of ϕ mapping function for Gene-Protein-Reaction rules in metabolic networks

As defined, the ϕ function maps gene expression data $\mathbf{D}$ onto a metabolic network $\mathcal{H}$ (where edges represent reactions) to produce an annotated network $\mathcal{H}'$ with data vectors $\mathbf{d}_e$ on its edges. Here, we detail how these edge data vectors, representing reaction scores, are computed using Gene-Protein-Reaction (GPR) rules.

For a reaction corresponding to an edge $e \in \mathcal{E}$ with a GPR rule, the score for this reaction in sample s_i, denoted $d_e^{(i)}$, is calculated by translating the GPR rule's boolean logic applied to the expression values of the relevant genes from $\mathbf{D}$ in sample s_i:

- **OR ($\vee$):** Maps to the maximum expression value. For genes $g_1 \vee \cdots \vee g_k$ in a GPR: $d_e^{(i)} = \max(\mathbf{D}_{g_1,i}, \ldots, \mathbf{D}_{g_k,i})$.
- **AND ($\wedge$):** Maps to the minimum expression value. For genes $g_1 \wedge \cdots \wedge g_k$ in a GPR: $d_e^{(i)} = \min(\mathbf{D}_{g_1,i}, \ldots, \mathbf{D}_{g_k,i})$.

Example Application

- **Input Data**:
  Assume we have gene expression data for three genes across two samples:

$$\mathbf{D} = \begin{array}{c|cc} & s_1 & s_2 \\ \hline g_1 & 2.0 & 2.5 \\ g_2 & 1.5 & 3.0 \\ g_3 & 0.5 & 0.7 \end{array}$$

  Here, each row corresponds to a gene, and each column corresponds to a sample.
- **Prior Knowledge Network (PKN)**:
  - **Metabolic Network $\mathcal{H} = (\mathcal{V}, \mathcal{E})$**:
    * Vertices $\mathcal{V}$: Metabolites.
    * Edges $\mathcal{E}$: Reactions between metabolites catalysed by the enzymes, which are linked to enzyme-coding genes through GPR rules.
- **GPR Rules**:
  - **GPR Rule for reaction 1 (associated with edge e_1)**:

$$\text{Reaction 1: } e_1 = g_1 \text{ OR } g_2$$

    The reaction occurs if either gene g_1 or g_2 is expressed. The edge e_1 will be assigned the maximum expression of g_1 or g_2.
  - **GPR Rule for reaction 2 (associated with edge e_2)**:

$$\text{Reaction 2: } e_2 = g_2 \text{ AND } g_3$$

    The reaction occurs only if both g_2 and g_3 are expressed. The edge e_2 will be assigned the minimum expression of g_2 and g_3.

- **Applying the Data Mapping Function ϕ**
  - **Calculate edge values**:
    - For edge e_1 (Reaction 1):

$$\mathbf{d}_{e_1} = \max(g_1, g_2)$$

      Across samples:

$$\mathbf{d}_{e_1} = [\max(2.0, 1.5), \max(2.5, 3.0)] = [2.0, 3.0]$$

    - For edge e_2 (Reaction 2):

$$\mathbf{d}_{e_2} = \min(g_2, g_3)$$

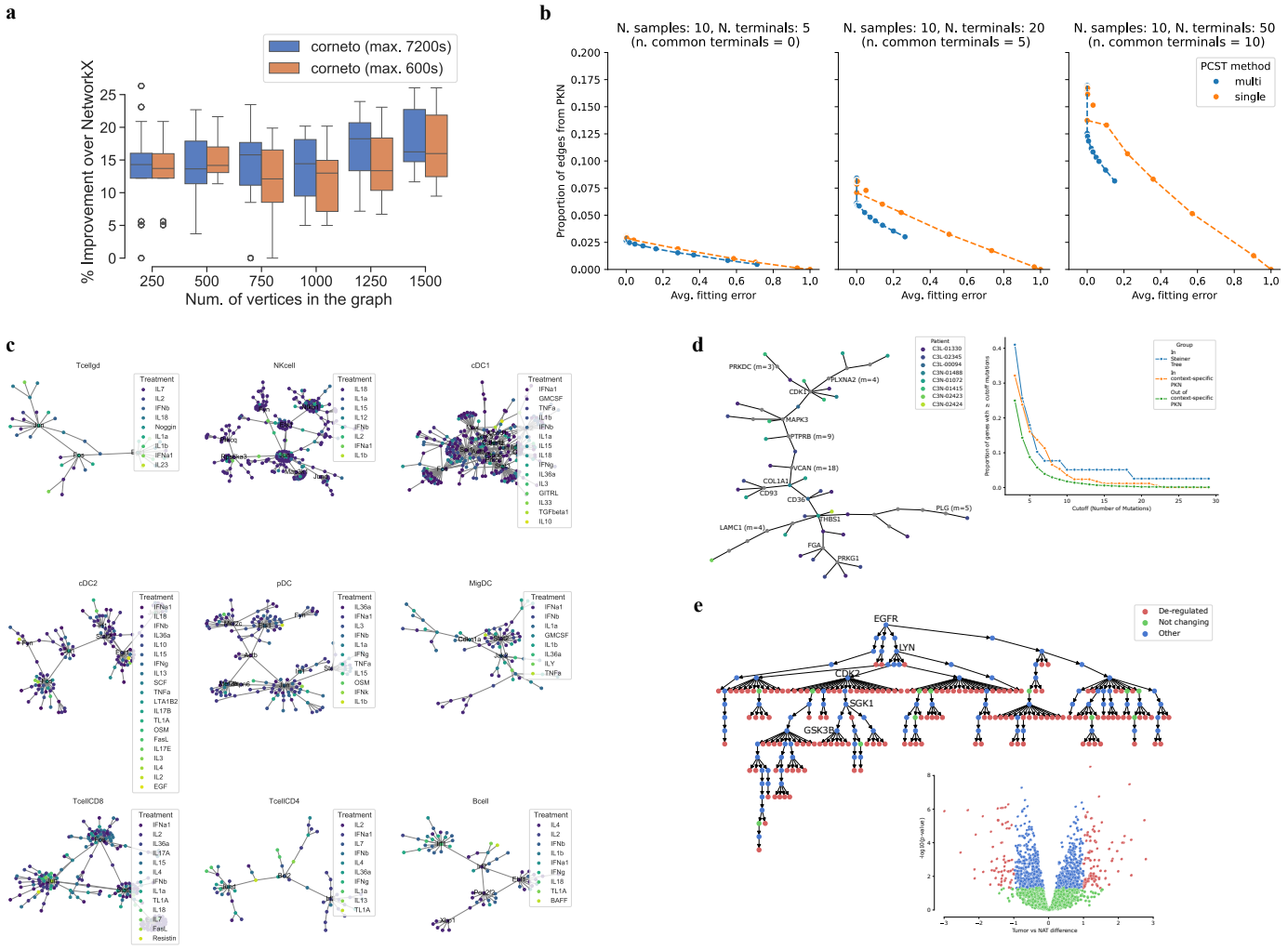
      Across samples:

$$\mathbf{d}_{e_2} = [\min(1.5, 0.5), \min(3.0, 0.7)] = [0.5, 0.7]$$

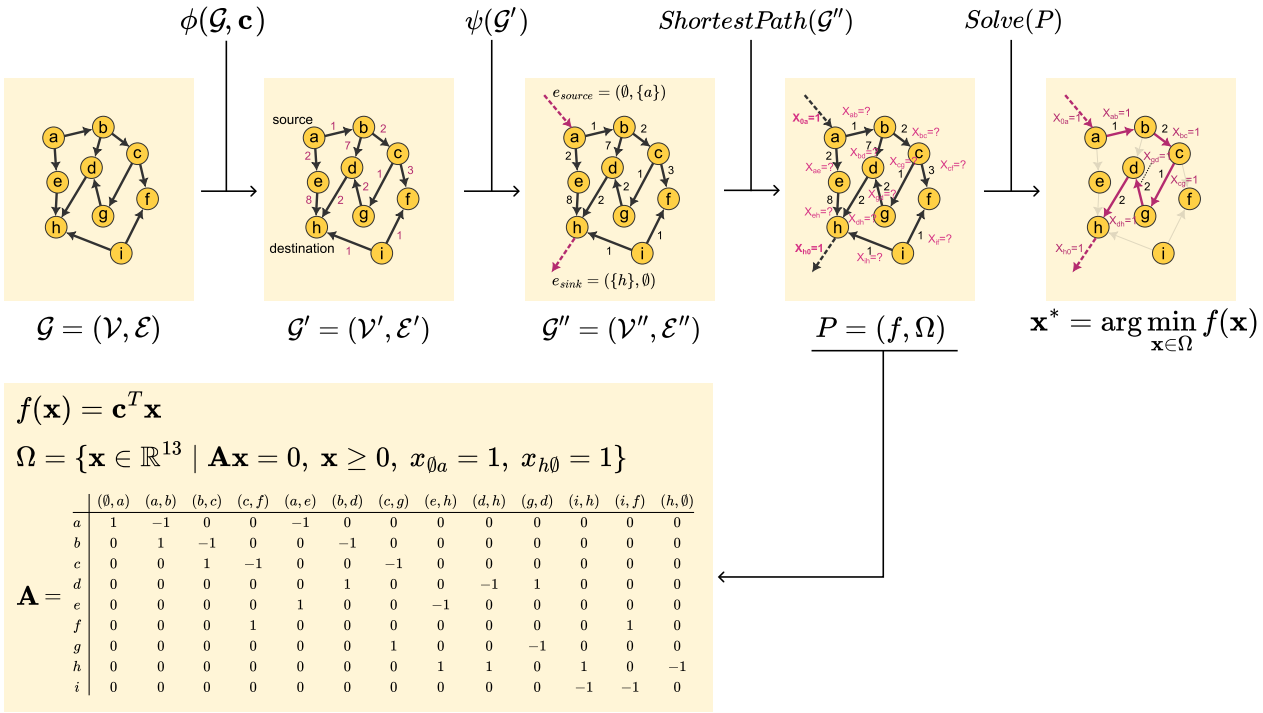
  - **Construct the annotated network**:
    - **Vertices** $\mathcal{V}'$: Since vertices do not have assigned values in this context, they remain unweighted.
    - **Edges** $\mathcal{E}'$: The edges are assigned the computed vectors based on the GPR rules:

$$\mathcal{E}' = \left\{ \begin{array}{l} e'_1 = (e_1, \mathbf{d}_{e_1} = [2.0, 3.0]) \,, \\ e'_2 = (e_2, \mathbf{d}_{e_2} = [0.5, 0.7]) \end{array} \right\}$$

Supplementary Figures

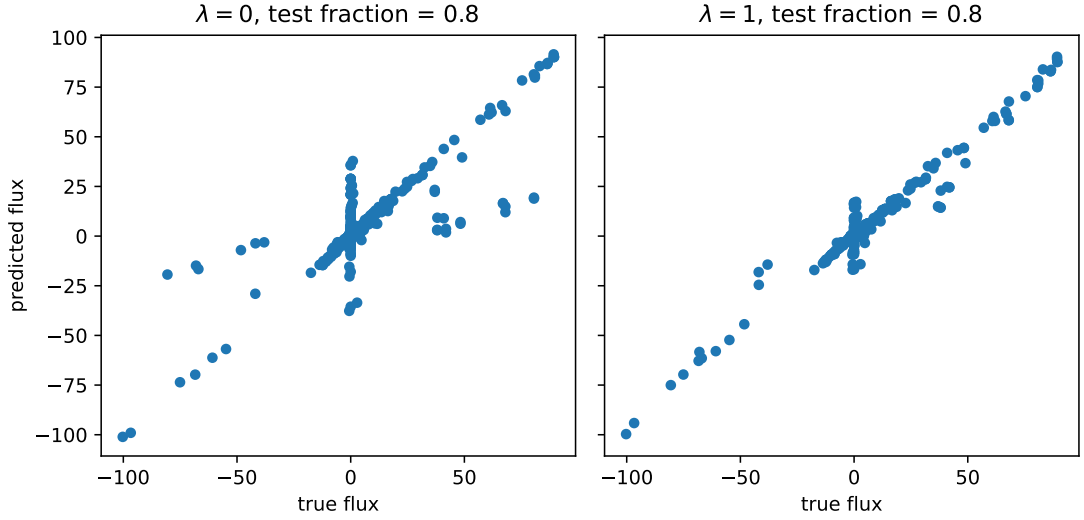


Supplementary Figure S1 Application of optimal single/multi-sample Steiner Trees on three different use cases
a, Performance comparison between the exact Steiner tree method in CORNETO and the Steiner tree method from the NetworkX python library. The bar plot shows the percentage improvement (y-axis) in the cost of Steiner trees generated by CORNETO compared to those from NetworkX across various random graphs (x-axis). CORNETO, which solves the exact problem, achieves an average cost reduction of approximately 15% on random graphs. This reduction increases with the number of vertices in the graph, as larger graphs contain more alternative paths. The box shows the distribution of relative improvement of CORNETO w.r.t NetworkX across the six graph-size categories under two time limits: 600 s ($n \in [61, 60, 60, 60, 60, 60]$ instances per size) and 7,200 s ($n \in [88, 88, 99, 99, 87, 89]$ solved instances for each graph size, respectively). Boxes in the box plot span the interquartile range (25th–75th percentiles), the center line marks the median (50th percentile), whiskers extend to the most extreme values within 1.5 × IQR of the quartiles, and points beyond are plotted individually as outliers. **b**, Simulation study to evaluate the performance of the multi-sample PCST vs single sample PCST. We generated three uses cases with different number of terminals and different number of common terminals across samples (where terminals are prized vertices). **c**, Context-specific molecular interaction networks from single-cell gene expression data using the Cytokine Atlas dataset [4]. Networks were inferred using multi-sample Steiner trees, where each sample corresponds to a treatment. For each cell type, we retrieve the 10 treatments with the highest number of differentially expressed genes. We then construct a multi-sample Steiner tree for each cell type using the deregulated genes from each top treatment as terminals. Networks represent the union of the steiner trees per celltype/treatment. **d**, Inferred network from proteomics data (CPTAC) for a cluster of 10 patients with similar absolute differential abundant proteomics profile (tumour vs normal). Labelled vertices with a number indicate the number of missense mutations in the LUAD cohort for that protein (only shown if ≥ 3 mutations). Labelled vertices without a number are hub nodes connecting at least 4 neighbors in the network. Right plot shows the proportion of proteins in each type of network (context-free, context-specific and PCST) that have at least N missense mutations, indicated in the x-axis. **e**, Application of directed PCST for integration of phosphoproteomics data from CPTAC Clear Cell Renal Cell Carcinoma study [3]. Data was processed with FragPipe Analyst. Phosphosites were classified into three groups: De-regulated (adjusted P value ≤ 0.01 and logFC ≥ 1), Other (adjusted P value ≤ 0.01 and logFC ≤ 1), and Not changing (adjusted P value ≥ 0.01). EGF receptor (EGFR) was used as the root node.

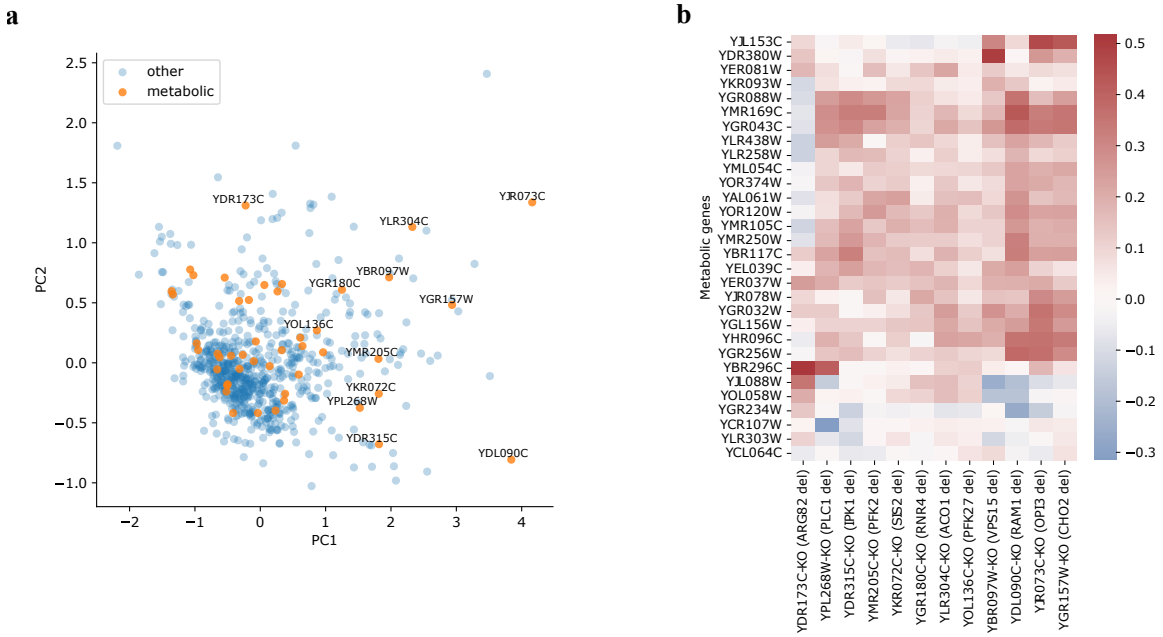


	$(\emptyset, a)$	(a, b)	(b, c)	(c, f)	(a, e)	(b, d)	(c, g)	(e, h)	(d, h)	(g, d)	(i, h)	(i, f)	$(h, \emptyset)$
a	1	−1	0	0	−1	0	0	0	0	0	0	0	0
b	0	1	−1	0	0	−1	0	0	0	0	0	0	0
c	0	0	1	−1	0	0	−1	0	0	0	0	0	0
d	0	0	0	0	0	1	0	0	−1	1	0	0	0
e	0	0	0	0	1	0	0	−1	0	0	0	0	0
f	0	0	0	1	0	0	0	0	0	0	0	1	0
g	0	0	0	0	0	0	1	0	0	−1	0	0	0
h	0	0	0	0	0	0	0	1	1	0	1	0	−1
i	0	0	0	0	0	0	0	0	0	0	−1	−1	0

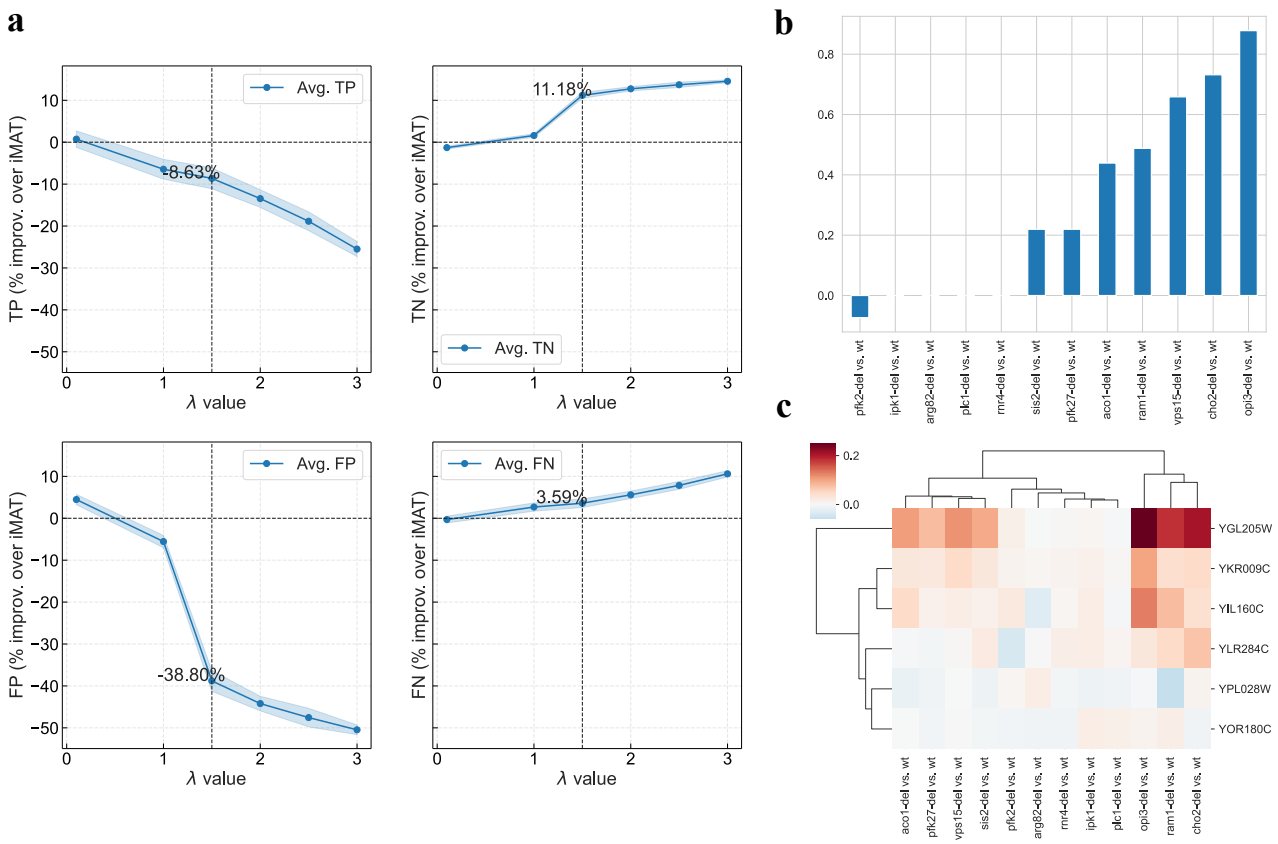
Supplementary Figure S2 Illustration of how the simple shortest-path problem can be implemented in CORNETO's framework. Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and some cost vector $\mathbf{c} \in \mathbb{R}^{|E|}$, the cost weights are mapped to the graph through the mapping function $\psi(\mathcal{G}, \mathbf{c})$, producing the graph $\mathcal{G}'$ where each edge is associated with a cost from $\mathbf{c}$, indicated with the red labels in the second step. Then, using the information about the source and destination vertices, the graph transformation function ϕ attaches an edge to inject flow through the source vertex (a) and one to extract flow through the destination vertex, to obtain a flow graph (third step). The *ShortestPath* method is implemented using CORNETO's framework. Flow variables and constraints are automatically created from the transformed graph, using its incidence matrix $\mathbf{A}$, and the objective function $\mathbf{c}^\top \mathbf{x}$, specified by the shortest path method. Finally, the problem is translated into a canonical form using one backend (e.g. CVXPY) and one of the compatible solvers computes the optimal solution and gives back the value for the flow variables, which define the shortest path.



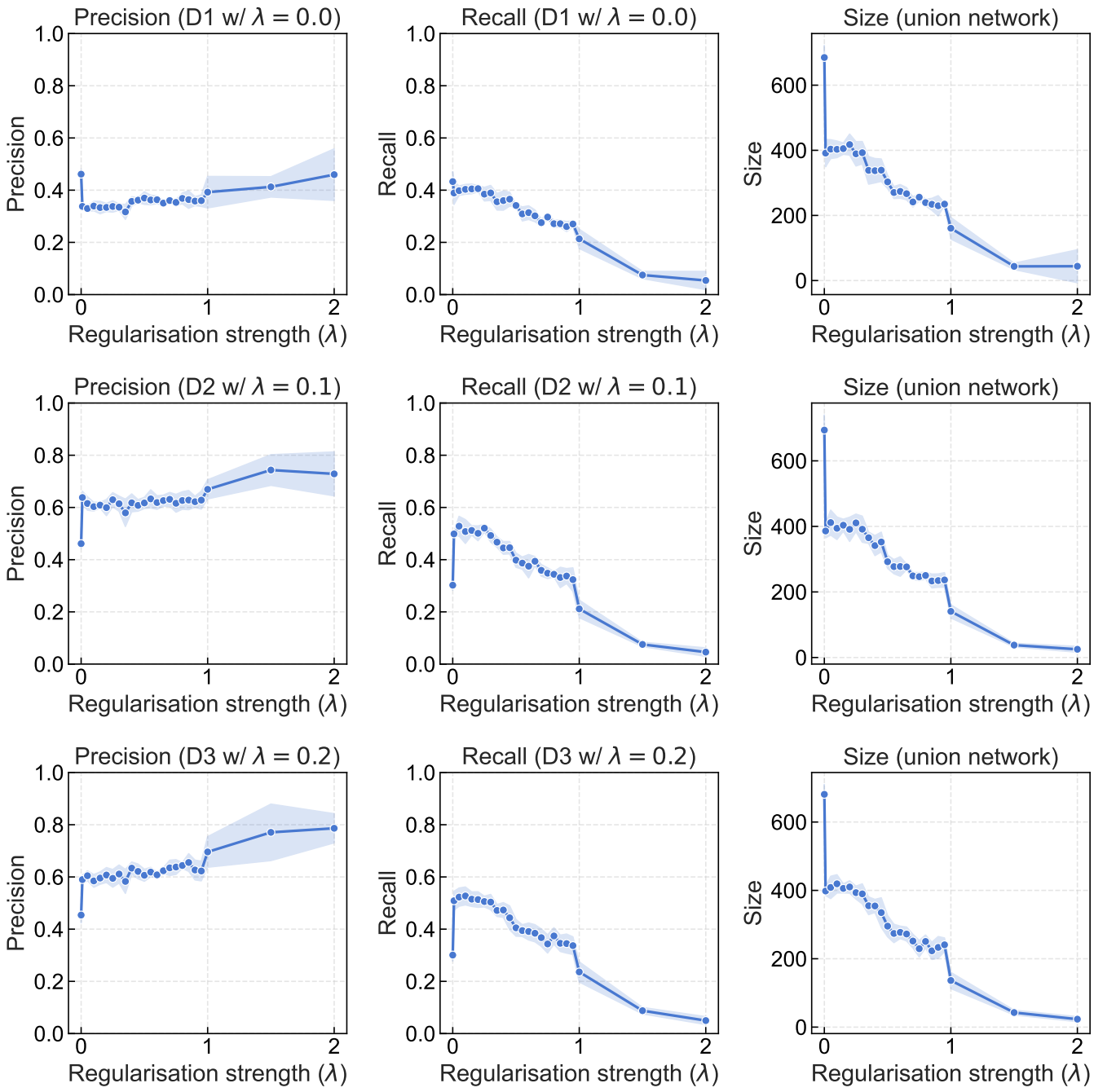
Supplementary Figure S3 True (simulated) fluxes vs. predicted fluxes with and without structured sparsity regularisation. The true fluxes were simulated with additive Gaussian noise (zero mean, standard deviation $\sigma = 1.0$), and 80% of the fluxes were used as test data. The left panel shows predictions without regularisation, while the right panel includes structured sparsity regularisation with $\lambda = 1$.



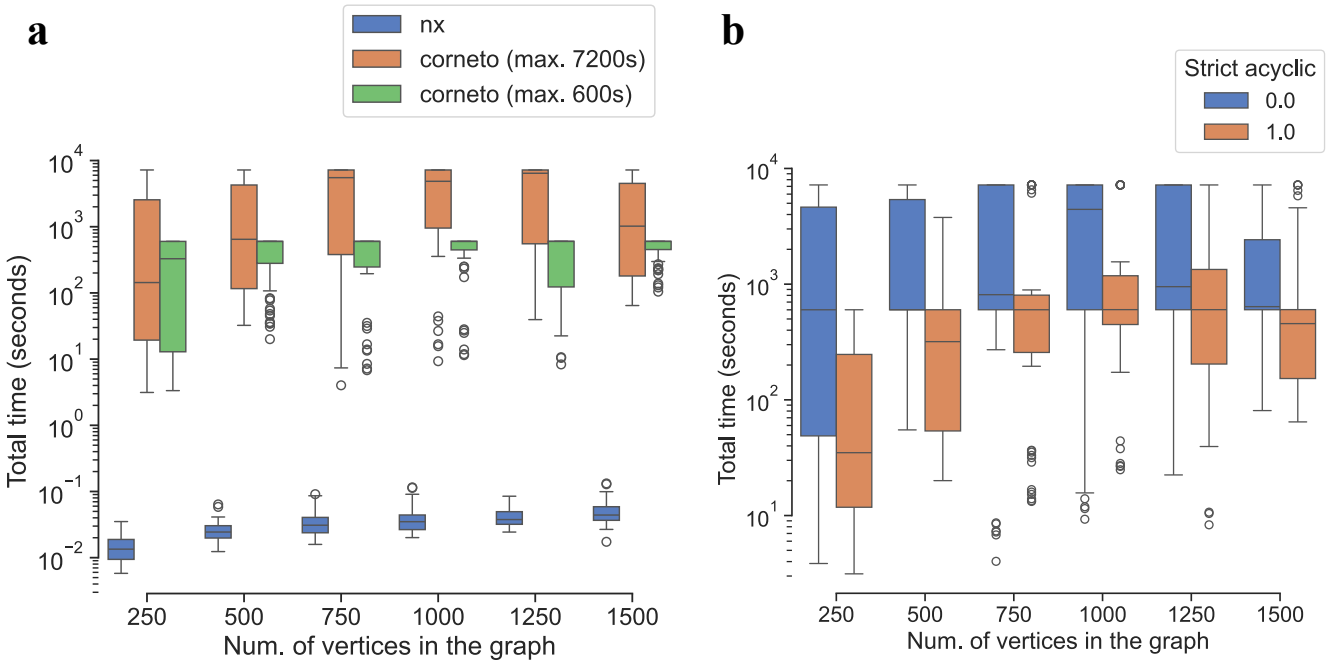
Supplementary Figure S4 **a**, Yeast gene deletion microarray dataset for validation, knockouts in the dataset are represented in a PCA plot, where dots in orange highlight the 12 selected samples for which the knocked-out gene corresponds to a metabolic-related gene present in the Yeast-GEM v8.5.0 PKN. **b**, Heatmap with the microarray gene expression (M/A ratios) for the top 30 most variable genes across the 12 selected samples



Supplementary Figure S5 **a**, Relative improvement of the CORNETO's multi-sample iMAT over the standard iMAT ($\lambda = 0$) for the true positives (top left), true negatives (top right), false positives (bottom left), and false negatives (bottom right). Shaded areas indicate $\pm 1\sigma$, calculated by re-running the method 10 times with different random seeds using Gurobi. **b**, Average reaction scores for the "Fatty acid degradation" pathway. PFK2, IPK1, ARG82, PLC1 and RNR4 KOs receive a score of 0 (unknown) for almost all reactions in the pathway, based on the transcriptomics data. **c**, Heatmap of M/A rations for the gene expression of the genes annotated in the "Fatty acid degradation" pathway, from which the reaction scores were calculated from.



Supplementary Figure S6 Simulation study using multi-sample CARNIVAL.
The goal of the evaluation is to measure the ability of the multi-sample method of recovering the original network when part of the input data (TFs) is hidden. Given the lack of ground truth, we generated three synthetic datasets (D1, D2, D3) by applying multi-sample CARNIVAL to the PANACEA dataset (Ponatinib treatment across 7 cell lines) under three different λ settings: 0 (first row), 0.1 (second row), and 0.2 (third row). To evaluate the method, we performed 5-fold cross-validation on the transcription factors across the 7 cell lines, for each dataset. In each fold, multi-sample CARNIVAL inferred the signalling networks from the held-out TFs, using the λ value indicated on the x-axis. The inferred networks were then converted into binary vectors (by concatenating data from all samples) and compared to the ground truth networks using precision and recall metrics. Error bands indicate $\pm 1\sigma$ of the five fold-wise results.



Supplementary Figure S7 Computation time for the Steiner tree simulation experiment.
a, Time in seconds (log scale) comparing the time of the heuristic method implemented in NetworkX and CORNETO. X-axis shows the number of vertices in the simulated graphs. The box shows the distribution across the six graph-size categories under two time limits for CORNETO: 600 s ($n \in [61, 60, 60, 60, 60, 60]$ instances per size) and 7,200 s ($n \in [88, 88, 99, 99, 87, 89]$ solved instances for each graph size, respectively. It also shows the distribution of time for NetworkX (nx), with no time limit, for the same instances used in CORNETO ($n \in [149, 148, 150, 150, 147, 149]$, respectively). **b**, Computation time using the option *strict acyclic* which forces that any feasible solution has to be a valid tree (Ω_{Tree}, Supplementary Methods). Although this constraint introduces more binary variables, the computation time is reduced as the space of feasible solutions is reduced from any sub-graph to only trees. Number of samples represented in the box plot: $n \in [74, 73, 75, 75, 72, 74]$ solved instances for strict acyclic = 0, and $n = 75$ solved instances per graph size, for strict acyclic = 1. Boxes span the interquartile range (25th–75th percentiles), the center line marks the median (50th percentile), whiskers extend to the most extreme values within 1.5 × IQR of the quartiles, and points beyond are plotted individually as outliers.

References

[1] Egon Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[2] V Bilim, A Ougolkov, K Yuuki, S Naito, H Kawazoe, A Muto, M Oya, D Billadeau, T Motoyama, and Y Tomita. Glycogen synthase kinase-3: a new therapeutic target in renal cell carcinoma. *British journal of cancer*, 101(12):2005–2014, 2009.

[3] David J Clark, Saravana M Dhanasekaran, Francesca Petralia, Jianbo Pan, Xiaoyu Song, Yingwei Hu, Felipe da Veiga Leprevost, Boris Reva, Tung-Shing M Lih, Hui-Yin Chang, et al. Integrated proteogenomic characterization of clear cell renal cell carcinoma. *Cell*, 179(4):964–983, 2019.

[4] Ang Cui, Teddy Huang, Shuqiang Li, Aileen Ma, Jorge L Pérez, Chris Sander, Derin B Keskin, Catherine J Wu, Ernest Fraenkel, and Nir Hacohen. Dictionary of immune responses to cytokines at single-cell resolution. *Nature*, 625(7994):377–384, 2024.

[5] Marcus T Dittrich, Gunnar W Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller. Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):i223–i231, 2008.

[6] Eugene F Douglass, Robert J Allaway, Bence Szalai, Wenyu Wang, Tingzhong Tian, Adrià Fernández-Torras, Ron Realubit, Charles Karan, Shuyu Zheng, Alberto Pessia, et al. A community challenge for a pancancer drug mechanism of action inference from perturbational profile data. *Cell Reports Medicine*, 3(1), 2022.

[7] Nikolaus Fortelny and Christoph Bock. Knowledge-primed neural networks enable biologically interpretable deep learning on single-cell sequencing data. *Genome biology*, 21:1–36, 2020.

[8] Michael A Gillette, Shankha Satpathy, Song Cao, Saravana M Dhanasekaran, Suhas V Vasaikar, Karsten Krug, Francesca Petralia, Yize Li, Wen-Wei Liang, Boris Reva, et al. Proteogenomic characterization reveals therapeutic vulnerabilities in lung adenocarcinoma. *Cell*, 182(1):200–225, 2020.

[9] Enio Gjerga, Aurelien Dugourd, Luis Tobalina, Abel Sousa, and Julio Saez-Rodriguez. Phonemes: efficient modeling of signaling networks derived from large-scale mass spectrometry data. *Journal of Proteome Research*, 20(4):2138–2144, 2021.

[10] Aric Hagberg, Pieter J Swart, and Daniel A Schult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.

[11] Rendy Hosea, Sharon Hillary, Sumera Naqvi, Shourong Wu, and Vivi Kasim. The two sides of chromosomal instability: drivers and brakes in cancer. *Signal transduction and targeted therapy*, 9(1):75, 2024.

[12] Shao-shan Carol Huang and Ernest Fraenkel. Integrating proteomic, transcriptional, and interactome data reveals hidden components of signaling and regulatory networks. *Science signaling*, 2(81):ra40–ra40, 2009.

[13] Frank K Hwang and Dana S Richards. Steiner tree problems. *Networks*, 22(1):55–89, 1992.

[14] Trey Ideker, Owen Ozier, Benno Schwikowski, and Andrew F Siegel. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, 18(suppl_1):S233–S240, 2002.

[15] Arthur B Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962.

[16] Patrick Kemmeren, Katrin Sameith, Loes AL Van De Pasch, Joris J Benschop, Tineke L Lenstra, Thanasis Margaritis, Eoghan O'Duibhir, Eva Apweiler, Sake van Wageningen, Cheuk W Ko, et al. Large-scale genetic perturbations reveal regulatory networks and an abundance of gene-specific repressors. *Cell*, 157(3):740–752, 2014.

[17] Yize Li, Yongchao Dou, Felipe Da Veiga Leprevost, Yifat Geffen, Anna P Calinawan, François Aguet, Yo Akiyama, Shankara Anand, Chet Birger, Song Cao, et al. Proteogenomic data and resources for pan-cancer analysis. *Cancer cell*, 41(8):1397–1406, 2023.

[18] Anika Liu, Panuwat Trairatphisan, Enio Gjerga, Athanasios Didangelos, Jonathan Barratt, and Julio Saez-Rodriguez. From expression footprints to causal pathways: contextualizing large signaling networks with carnival. *NPJ systems biology and applications*, 5(1):40, 2019.

[19] Wout Megchelenbrink, Martijn Huynen, and Elena Marchiori. optgpsampler: an improved tool for uniformly sampling the solution-space of genome-scale metabolic networks. *PloS one*, 9(2):e86587, 2014.

[20] Alexander Mitsos, Ioannis N Melas, Paraskeuas Siminelakis, Aikaterini D Chairakaki, Julio Saez-Rodriguez, and Leonidas G Alexopoulos. Identifying drug effects via pathway alterations using an integer linear programming optimization formulation on phosphoproteomic data. *PLoS computational biology*, 5(12):e1000591, 2009.

[21] Victor Paton, Ricardo Omar Ramirez Flores, Attila Gabor, Pau Badia-i Mompel, Jovan Tanevski, Martin Garrido-Rodriguez, and Julio Saez-Rodriguez. Assessing the impact of transcriptomics data analysis pipelines on downstream functional enrichment results. *Nucleic Acids Research*, 52(14):8100–8111, 2024.

[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[23] Pablo Rodriguez-Mier, Louison Fresnais, Nathalie Poupin, and Fabien Jourdan. Miom: Mixed integer optimization for metabolism. https://github.com/MetExplore/miom, 2021.

[24] Arie Segev. The node-weighted steiner tree problem. *Networks*, 17(1):1–17, 1987.

[25] Yuan-Xiang Shi, Tao Zhu, Ting Zou, Wei Zhuo, Yi-Xin Chen, Ma-Sha Huang, Wei Zheng, Chen-Jing Wang, Xi Li, Xiao-Yuan Mao, et al. Prognostic and predictive values of cdk1 and mad2l1 in lung adenocarcinoma. *Oncotarget*, 7(51):85235, 2016.

[26] Tomer Shlomi, Moran N Cabili, Markus J Herrgård, Bernhard Ø Palsson, and Eytan Ruppin. Network-based prediction of human tissue-specific metabolism. *Nature biotechnology*, 26(9):1003–1010, 2008.

[27] Anthony C Smith, Filmon Eyassu, Jean-Pierre Mazat, and Alan J Robinson. Mitocore: a curated constraint-based model for simulating human central metabolism. *BMC systems biology*, 11:1–13, 2017.

[28] Benjamín Sánchez, feiranl, Eduard Kerkhoven, eiden309, Mihail Anton, William Scott, Thomas Pfau, hongzhonglu, Dimitra Lappa, Sebastián Mendoza Farías, and The Gitter Badger. Sysbiochalmers/yeast-gem: yeast 8.5.0, July 2021.

[29] Camille DA Terfve, Edmund H Wilkes, Pedro Casado, Pedro R Cutillas, and Julio Saez-Rodriguez. Large-scale models of signal propagation in human cells derived from discovery phosphoproteomic data. *Nature communications*, 6(1):8033, 2015.

[30] Nurcan Tuncbag, Sara JC Gosline, Amanda Kedaigle, Anthony R Soltis, Anthony Gitter, and Ernest Fraenkel. Network-based interpretation of diverse high-throughput datasets through the omics integrator software package. *PLoS computational biology*, 12(4):e1004879, 2016.

[31] Dénes Türei, Tamás Korcsmáros, and Julio Saez-Rodriguez. Omnipath: guidelines and gateway for literature-curated signaling pathway resources. *Nature methods*, 13(12):966–967, 2016.

[32] Tzu-Yang Weng, Chih-Yang Wang, Yu-Hsuan Hung, Wei-Ching Chen, Yi-Ling Chen, and Ming-Derg Lai. Differential expression pattern of thbs1 and thbs2 in lung cancer: clinical outcome and a systematic-analysis of microarray databases. *PloS one*, 11(8):e0161007, 2016.

[33] Fengchao Yu, Guo Ci Teo, Andy T Kong, Klemens Fröhlich, Ginny Xiaohe Li, Vadim Demichev, and Alexey I Nesvizhskii. Analysis of DIA proteomics data using MSFragger-DIA and FragPipe computational platform. *Nat. Commun.*, 14(1):4154, July 2023.