Contextualizing large scale signalling networks from expression footprints with CARNIVAL

Enio Gjerga *[1,2], Panuwat Trairatphisan[2], Anika Liu[2], Alberto Valdeolivas[2], and Nicolas Peschke[2]

[1]RWTH Aachen University, Faculty of Medicine, Joint Research Centre for Computational Biomedicine (JRC-COMBINE), 52074, Aachen, Germany
[2]Heidelberg University, Faculty of Medicine, and Heidelberg University Hospital, Institute of Computational Biomedicine, Bioquant, 69120, Heidelberg, Germany

March 26, 2020

Contents

1 Introduction

While gene expression profiling is commonly used to gain an overview of cellular processes, the identification of upstream processes that drive expression changes remains a challenge. To address this issue, we introduce *CARNIVAL* [1], a causal network contextualization tool which derives network architectures from gene expression footprints. *CARNIVAL* (CAusal Reasoning pipeline for Network identification using Integer VALue programming)(see `https://saezlab.github.io/CARNIVAL/`) integrates different sources of prior knowledge including signed and directed protein–protein interactions, transcription factor targets, and pathway signatures.

1.1 CARNIVAL pipeline

CARNIVAL refines a quantitative objective function for ILP problem by incorporating TF and pathway activities on a continuous scale. In addition, the CARNIVAL framework allows us to contextualize the network with or without known targets of perturbations. The implementation is separated into two pipelines

*enio.gjerga@gmail.com

which will be referred henceforth as Standard CARNIVAL *StdCARNIVAL* (with known perturbation targets as an input) and Inverse CARNIVAL *InvCARNIVAL* (without information on targets of perturbation), see Figure 1. The differential gene expression is used to infer transcription factor (TF) activities with DoRothEA, which are subsequently discretized in order to formulate ILPconstraints. As a result, CARNIVAL derives a family of highest scoring networks which best explain theinferred TF activities. Continuous pathway and TF activities can be additionally considered in the objective function.

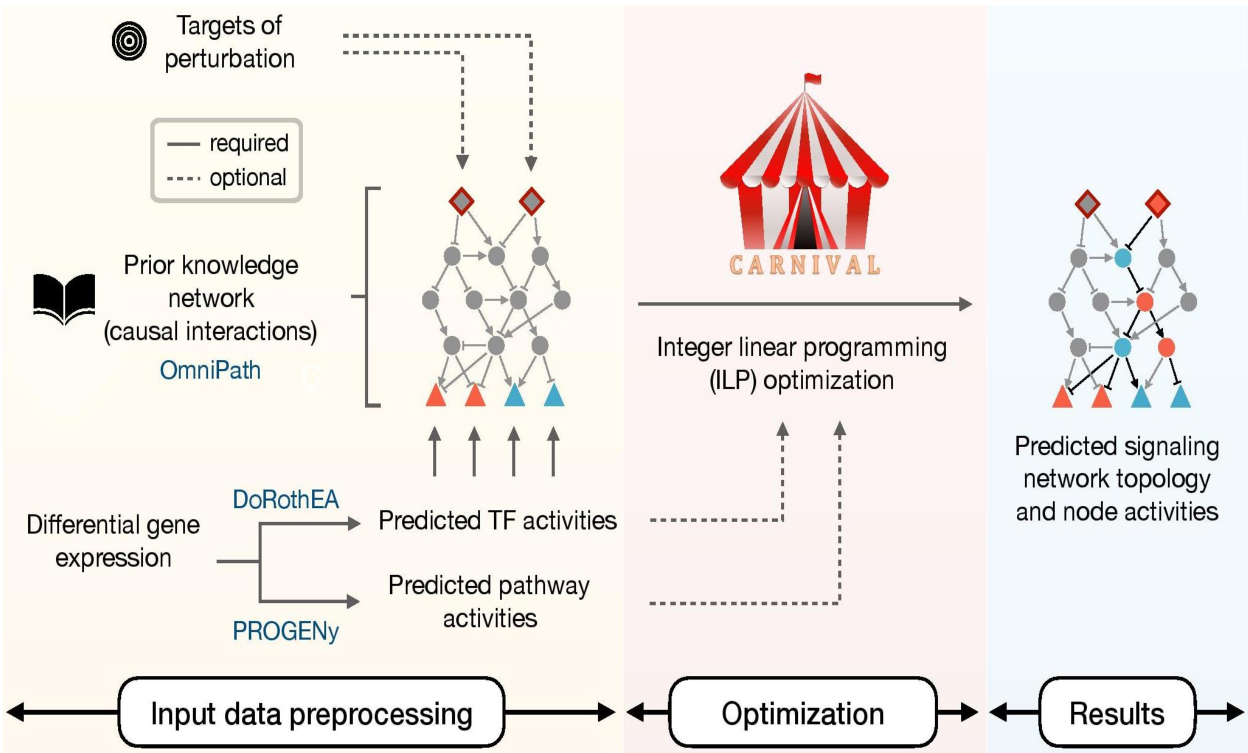

Figure 1: *CARNIVAL* pipeline

1.2 ILP solvers

CARNIVAL is an extension of the previously implemented Causal Reasoning method from Melas et al. [2].The network inference process is swiftly performed with an Integer Linear Programming (ILP) formulation of causal reasoning using three solvers: the *R-CRAN lpSolve* free software used for solving linear problems; the open-source mixed integer programming solver *Cbc* (Coin-or branch and cut)(see `https://projects.coin-or.org/Cbc`); or the *CPLEX optimizer* from *IBM* (see `https://www.ibm.com/analytics/cplex-optimizer`) which can be obtained for free through the Academic Initiative. To perform the analysis with *cplex* or *cbc*, the users will then need to store the binary *cbc* or *cplex* executables on any directory they wish. The binary files of *cbc* can be found by first downloading one of the optimization suites provided here: `https://www.coin-or.org/download/binary/OptimizationSuite/`, unzip the download and from there save the *cbc* executable (which can be found on the *bin* directory) file on any of the direcotries they wish of their machines. As for the *cplex*, the executable file can be obtained after registration on the *ILOG CPLEX Optimization Studio* here: `https://my15.digitalexperience.ibm.com/b73a5759-c6a6-4033-ab6b-d9d4f9a6d65b/dxsites/151914d1-03d2-48fe-97d9-d21166848e65/technology/data-science`. Similar like before, users will have to find the *cplex* executable binary file and save on a directory of their own wish or keep them on their default installation paths. The path to interactive version of *CPLEX* is differed based on the operating system. The default installation path for each OS is as follows:

For Mac OS:

```
# ~/Applications/IBM/ILOG/CPLEX_Studio129/cplex/bin/x86-64_osx/cplex
```

For Linux:

```
# /opt/ibm/ILOG/CPLEX_Studio129/cplex/bin/x86-64_linux/cplex
```

For Windows:

```
# C:/Program Files/IBM/ILOG/CPLEX_Studio129/cplex/bin/x64_win64/cplex.exe
```

Note that the version of *CPLEX* has to be changed accordingly (the latest current version is CPLEX-Studio129).

The *lpSolve* solver can be used after downloading and installing the *lpSolve* R-package (see https://cran.r-project.org/web/packages/lpSolve/index.html). This solver only works for smaller examples and it can give only one optimal solution. For larger real-case examples, the users can use *cbc* or *cplex* solvers.

While *Cbc* is open-source and can be used from any user, the *CPLEX* solver is more computationally efficient and is able to provide multiple equivalent solutions which are then combined. The *mipGAP*, *limitPop*, *poolCap*, *poolIntensity* and *poolReplace* work only if *CPLEX* solver is used to train the networks.

1.3 Citation

CARNIVAL can be cited as follows:

Liu, A., Trairatphisan, P., Gjerga, E. et al. From expression footprints to causal pathways: contextualizing large signaling networks with CARNIVAL. npj Syst Biol Appl 5, 40 (2019) doi:10.1038/s41540-019-0118-z

1.4 Prerequisites

Besides the above mentioned solvers, users need also to install the following R-package dependencies: *readr*(see https://cran.r-project.org/web/packages/readr/index.html); *igraph* (see https://igraph.org/r/); *readxl*(see https://readxl.tidyverse.org/); *dplyr*(see https://www.rdocumentation.org/packages/dplyr/versions/0.7.8); *lpSolve*(see https://cran.r-project.org/web/packages/lpSolve/index.html)

In order to visualize the automatically generated *CARNIVAL* networks, users will also need to download and install the Graph Visualization software *graphviz*(see https://www.graphviz.org/).

2 Running CARNIVAL

In the CARNIVAL package, built-in examples are available as the test cases as follows:

1. A small toy example where the inputs are known (*stdCARNIVAL*)
2. A small toy example where the inputs are not known (*invCARNVAL*)

2.1 Toy Example - 1

Let us consider the toy example of 2. In this case, the inputs are known (*I1=1* and *I2=1*), while the network has all the interactions as activatory besides the connection connecting *I2* with *N2*. The measurements we have them both *M1=1* and *M2=1*.

Users can run the CARNIVAL analysis with the free solver as follows:

```
library(CARNIVAL)
load(file = system.file("toy_inputs_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_measurements_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_network_ex1.RData",
                        package="CARNIVAL"))
# lpSolve
result = runCARNIVAL(inputObj = toy_inputs_ex1, measObj = toy_measurements_ex1,
                     netObj = toy_network_ex1)
```

```
[1] "Writing constraints..."
[1] "Solving LP problem..."
```

```
print(result)
```

```
$weightedSIF
     Node1 Sign Node2 Weight
[1,] "I1"  "1"  "N1"  "100"
[2,] "N1"  "1"  "M1"  "100"
[3,] "N1"  "1"  "M2"  "100"

$nodesAttributes
     Node ZeroAct UpAct DownAct AvgAct NodeType
[1,] "I1" "0"     "100" "0"     "100"  "S"
[2,] "N1" "0"     "100" "0"     "100"  ""
[3,] "I2" "0"     "100" "0"     "100"  "S"
[4,] "N2" "100"   "0"   "0"     "0"    ""
[5,] "M1" "0"     "100" "0"     "100"  "T"
[6,] "M2" "0"     "100" "0"     "100"  "T"

$sifAll
$sifAll[[1]]
     Node1 Sign Node2
[1,] "I1"  "1"  "N1"
[2,] "N1"  "1"  "M1"
[3,] "N1"  "1"  "M2"
```

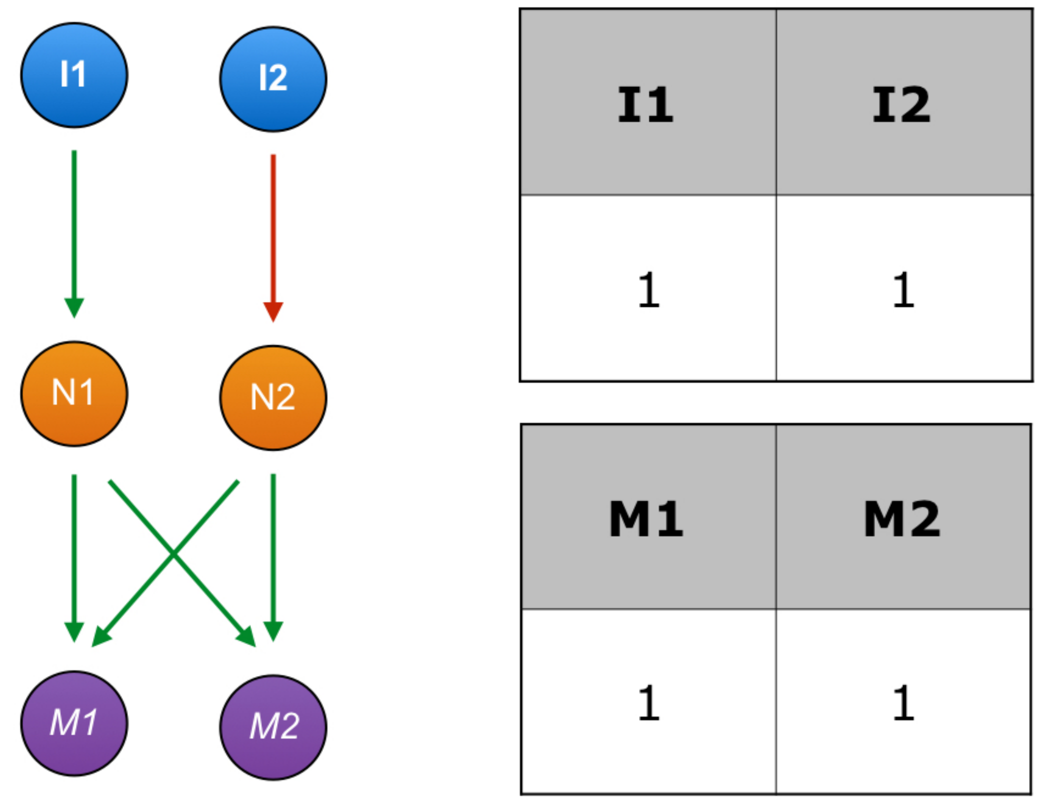

Figure 2: Prior Knowledge Network of Toy Example - 1

```
$attributesAll
$attributesAll[[1]]
     Nodes Activity
[1,] "I1"  "1"
[2,] "N1"  "1"
[3,] "I2"  "1"
[4,] "M1"  "1"
[5,] "M2"  "1"
```

The *result* object will contain the following fields:

- *weightedSIF*: which contains all the combined solutions generated (*lpSolve* and *cbc* will generate only 1 solution, while *cplex* can generate multiple solutions) with the Weight column indicating how frequently an interaction has appeard on the combined solution.

- *nodesAttributes*: indicating the weigted mean activities of each of the proteins present in the combined solution and the type of the node (Input (S), Measured (T) or Inferred).

- *sifAll*: A list containing all the separate CARNIVAL solutions.

The CARNIVAL results for the Toy Example - 1 are as shown in 3.

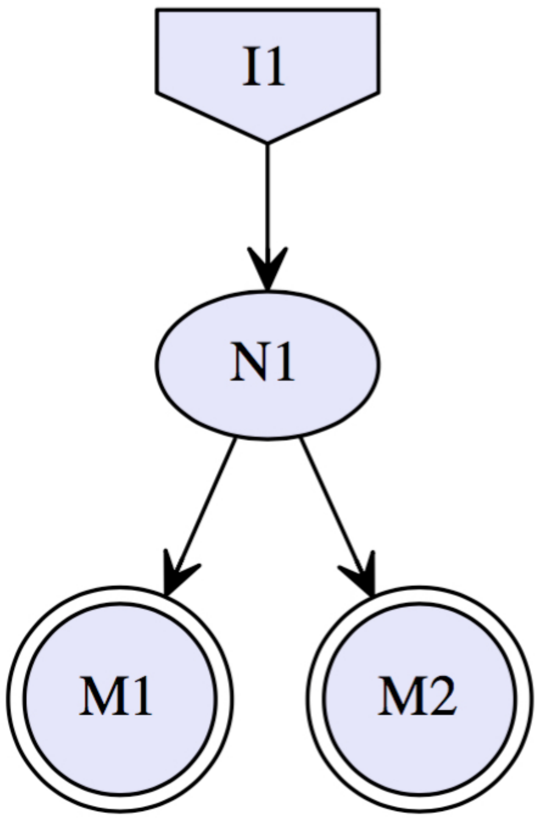

Figure 3: Solution network of Toy Example - 1

2.2 Toy Example - 2

Now let us consider the toy example of 4. In this case, the inputs are not known and thus a dummy *Perturbation* node will be automatically generated and which connects all the nodes in the network which do not have any incoming interaction via activatory and inhibitory interactions. In this way, CARNIVAL will infere the possible activities of the upper nodes in a way which best fits the downstram measurements

Users on this case can run the invCARNIVAL analysis with the free solver as follows:

```
library(CARNIVAL) # load CARNIVAL library
load(file = system.file("toy_measurements_ex2.RData",
                         package="CARNIVAL"))
load(file = system.file("toy_network_ex2.RData",
                         package="CARNIVAL"))
# lpSolve
result = runCARNIVAL(measObj = toy_measurements_ex2, netObj = toy_network_ex2)
```

```
[1] "inputObj set to NULL -- running InvCARNIVAL"
[1] "Writing constraints..."
[1] "Solving LP problem..."
```

```
print(result)
```

```
$weightedSIF
     Node1          Sign Node2 Weight
[1,] "I2"           "1"  "N1"  "100"
[2,] "I2"           "1"  "N2"  "100"
[3,] "N1"           "1"  "M1"  "100"
[4,] "N1"           "1"  "M2"  "100"
[5,] "N2"           "1"  "M2"  "100"
[6,] "N2"           "1"  "M3"  "100"
[7,] "Perturbation" "1"  "I2"  "100"

$nodesAttributes
      Node           ZeroAct UpAct DownAct AvgAct NodeType
 [1,] "I1"           "100"   "0"   "0"     "0"    ""
 [2,] "I2"           "0"     "100" "0"     "100"  ""
 [3,] "I3"           "100"   "0"   "0"     "0"    ""
 [4,] "N1"           "0"     "100" "0"     "100"  ""
 [5,] "N2"           "0"     "100" "0"     "100"  ""
 [6,] "Perturbation" "0"     "100" "0"     "100"  "S"
 [7,] "M1"           "0"     "100" "0"     "100"  "T"
 [8,] "M2"           "0"     "100" "0"     "100"  "T"
 [9,] "M3"           "0"     "100" "0"     "100"  "T"

$sifAll
$sifAll[[1]]
     Node1          Sign Node2
[1,] "I2"           "1"  "N1"
[2,] "I2"           "1"  "N2"
[3,] "N1"           "1"  "M1"
[4,] "N1"           "1"  "M2"
```

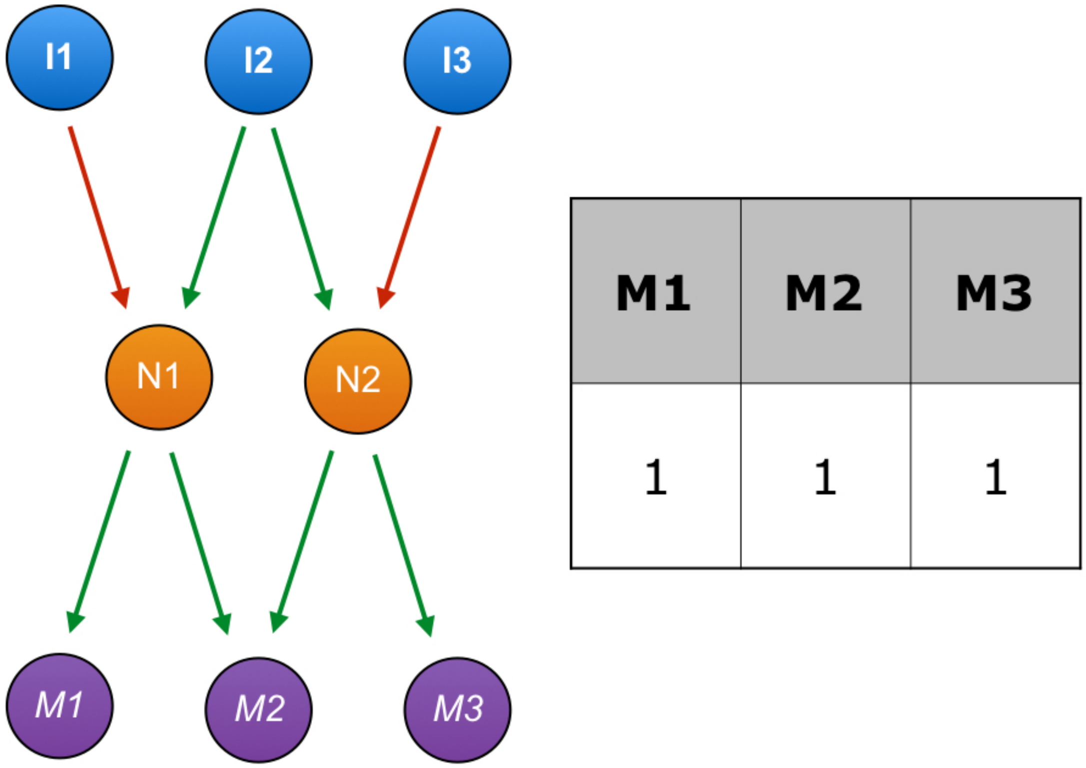

Figure 4: Prior Knowledge Network of Toy Example - 2

```
[5,] "N2"           "1"  "M2"
[6,] "N2"           "1"  "M3"
[7,] "Perturbation" "1"  "I2"


$attributesAll
$attributesAll[[1]]
     Nodes          Activity
[1,] "I2"           "1"
[2,] "N1"           "1"
[3,] "N2"           "1"
[4,] "Perturbation" "1"
[5,] "M1"           "1"
[6,] "M2"           "1"
[7,] "M3"           "1"
```

The CARNIVAL results for the Toy Example - 2 are as shown in 5.

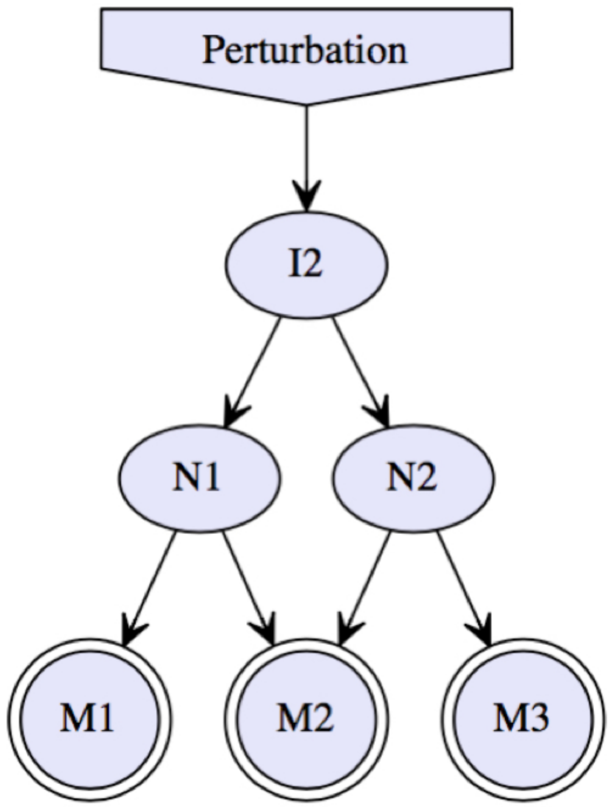

Figure 5: Solution network of Toy Example - 2

2.3 Real Case Example

Now we show an example about how we can run CARNIVAL on a real case-study. We will show step-by-step how we can build a prior knowledge of signalling and a DoRothEA regulon list from *OmniPath* [5] (see https://github.com/saezlab/OmnipathR for the R-package). Next we show how we can esti mate pathway activities via PROGENy (see https://github.com/saezlab/progeny for the R-package). Next we will demonstrate how we can use the *viper* [6] (see https://www.bioconductor.org/packages/release/bioc/html/viper.html for the R-package). Finally we will show how we can use CARNIVAL to combine all this information in order to infer the causal regulatory network.

Through *OmnipathR* package, we build the network object needed for contextualizing the signalling network with CARNIVAL.

For the analysis we consider the expression data example from the *progeny* R-package (see https://github.com/saezlab/progeny). We estimate normalized (from -1 to 1) pathway activity scores via the *progeny* function for the first sample and then we assign the inferred activities as node weights to the progeny members from the *progenyMembers* object loaded from *CARNIVAL* package. Users can estimate the pathway activities for any sample they wish (see documentation of *assignPROGENyScores()* function)

Next we can retrieve regulons from OmniPath and create the regulon table. From there we obtain the *viper* regulon list through the *createRegulonList* function (see documentation for more details).

From here we can estimate the TF activities and generate the continuos measurement inputs for CARNIVAL.

So far we have generated the prior knowledge network which will be used to contextualize the regulatory signalling network (*netObj*) as well as a list object containing the progeny weights (*weightObj*) and the TF activities (*tfList* to be assigned to the *measObj*). For running CARNIVAL we have to access one element at the time from the *weightObj* and *tfList*. We use all this as inputs to run the CARNIVAL analysis through the *runCARNIVAL* function. Since the input targets are not known, we perform the *invCARNIVAL* analysis.

```
##users need to define the path to the cplex solver

res <- runCARNIVAL(measObj = tfList[[1]], netObj = netObj,
                   weightObj = weightObj[[1]], solverPath = solverPath,
                   solver = "cplex", DOTfig = TRUE)
```

The solution network of this small real-case application is as shown in 6.

References

[1] A. Liu, P. Trairatphisan, E. Gjerga, A. Didangelos, J. Barratt and J. Saez-Rodriguez. From expression footprints to causal pathways: contextualizing large signaling networks with CARNIVAL. npj Syst Biol Appl 5, 40 (2019) doi:10.1038/s41540-019-0118-z.

[2] I.N. Melas, T. Sakellaropoulos, F. Iorio, L.G. Alexopoulos, W. Loh, D.A. Lauffenburger, J. Saez-Rodriguez and J.P.F. Bai Identification of drug-specific pathways based on gene expression data: application to drug induced lung injury. Integr Biol (Camb). 2015 Aug;7(8):904-20. doi: 10.1039/c4ib00294f.

[3] L. Garcia-Alonso, F. Iorio, A. Matchan, N. Fonseca, P. Jaaks, G. Peat, M. Pignatelli, F. Falcone, C.H. Benes, I. Dunham, G.R. Bignell, S. McDade, M.J. Garnett and J. Saez-Rodriguez Transcription Factor Activities Enhance Markers of Drug Sensitivity in Cancer. Cancer Research, 78(3), 769–780.

[4] M. Schubert, B. Klinger, M. Klunemann, A. Sieber, F. Uhlitz, S. Sauer, M.J. Garnett, N. Bluthgen and J. Saez-Rodriguez Perturbation-response genes reveal signaling footprints in cancer gene expression Nat Commun. 2018;9(1):20.

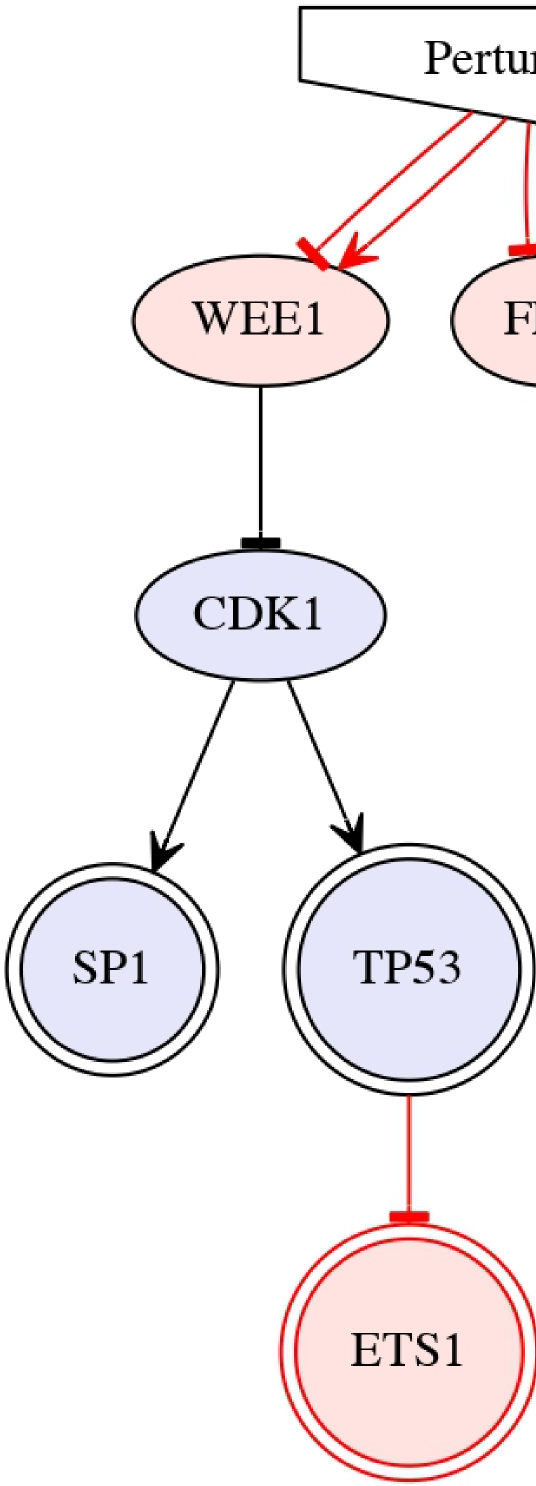

Figure 6: Solution for the real case example

[5] D. Turei, T. Korcsmáros, and J. Saez-Rodriguez OmniPath: guidelines and gateway for literature-curated signaling pathway resources. Nat Methods 2016 Nov 29;13(12):966-967

[6] M.J. Alvarez, Y. Shen, F.M. Giorgi, A. Lachman, B.B. Ding, B.H. Ye and A. Califano Functional characterization of somatic mutations in cancer using network-based inference of protein activity Nat Genetics volume 48, pages838–847(2016)