Subset Selection for Forecast Reconciliation

Xiaoqian Wang

2023-03-26

1 Hierarchical time series

Create a hierarchical structure as complete as possible. Only nested structure is considered here. Then the summing matrix $\boldsymbol{S}$ is given and $\hat{\boldsymbol{y}}_h$ can be obtained easily by specifying forecasting models.

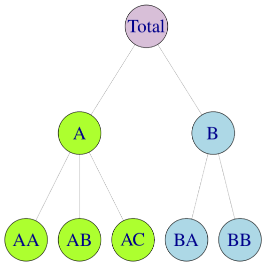


Figure 1: A 2-level hierarchical tree structure

For example,

$$\boldsymbol{S} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \hat{\boldsymbol{y}}_h = (10, 6, 5, 1, 4, 0, 2, 5)'.$$

2 Linear forecast reconciliation

$$\tilde{\boldsymbol{y}}_h = \boldsymbol{S}\boldsymbol{G}_h\hat{\boldsymbol{y}}_h,$$

where $\boldsymbol{G}_h$ maps the base forecasts $\hat{\boldsymbol{y}}_h$ into the bottom level, it combines all base forecasts to form bottom-level forecasts.

- **Bottom-up approach:** $\boldsymbol{G}_{bu} = \begin{bmatrix} \boldsymbol{O}_{n_b \times n_a} & \boldsymbol{I}_{n_b} \end{bmatrix}$

- **Top-down approach:** $\boldsymbol{G}_{td} = \begin{bmatrix} \boldsymbol{p} & \boldsymbol{O}_{n_b \times (n-1)} \end{bmatrix}$, where $\boldsymbol{p}$ is an n_b-dimensional vector including the set of disaggregation proportions.
- **Middle-out approach:** $\boldsymbol{G}_{mo} = \begin{bmatrix} \boldsymbol{O}_{n_b \times n_t} & \boldsymbol{P}_{n_b \times n_l} & \boldsymbol{O}_{n_b \times n_d} \end{bmatrix}$, where $n_t + n_l + n_d = n$, $n_d \geq n_b$, $n_t \geq 1$, $\sum_{i=1}^{n_b} \sum_{j=1}^{n_l} p_{ij} = n_l$.
- **Optimization approaches:** $\boldsymbol{G}_h = \left(\boldsymbol{S}'\boldsymbol{W}_h^{-1}\boldsymbol{S}\right)^{-1} \boldsymbol{S}'\boldsymbol{W}_h^{-1}$ is obtained by solving the optimization problem

$$\min_{\boldsymbol{G}_h} \left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}_h\hat{\boldsymbol{y}}_h\right)' \boldsymbol{W}_h^{-1} \left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}_h\hat{\boldsymbol{y}}_h\right), \text{ s.t. } \boldsymbol{G}_h\boldsymbol{S} = \boldsymbol{I}_{n_b}$$

3 Subset selection

a. About $\boldsymbol{G}_h$. Our goal is to **zero out some columns** so that the corresponding base forecasts in $\hat{\boldsymbol{y}}_h$ are not used to form the reconciled bottom-level forecasts and, furthermore, all reconciled forecasts.

b. About $\boldsymbol{S}$. It sums up the reconciled bottom-level forecasts to get the full set of reconciled forecasts. For the purpose of automatic selection and comparison, we don't need to zero out the corresponding rows.

We have to add additional constraints on $\boldsymbol{G}_h$. This leads to a main question:

How to mathematically describe the constraint?

3.1 Ideas based on MinT

3.1.1 Idea 1

$$\tilde{\boldsymbol{y}}_h = \boldsymbol{S}\boldsymbol{G}_h\boldsymbol{B}_h\hat{\boldsymbol{y}}_h,$$

where $\boldsymbol{B}_h$ is an $n \times n$ diagonal matrix with elements of the main diagonal being either zero or one.

The optimization problem can be written as

$$\begin{aligned} \min\nolimits_{\boldsymbol{G}_h} \left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}_h\boldsymbol{B}_h\hat{\boldsymbol{y}}_h\right)' \boldsymbol{W}_h^{-1} \left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}_h\boldsymbol{B}_h\hat{\boldsymbol{y}}_h\right) + \lambda \|\boldsymbol{B}_h\boldsymbol{1}\|_1, \\ \text{s.t. } \boldsymbol{G}_h\boldsymbol{B}_h\boldsymbol{S} = \boldsymbol{I}_{n_b} \\ b_{ii}(1 - b_{ii}) = 0 \text{ for } i = 1, 2, \ldots, n \\ b_{ij} = 0 \text{ for } i \neq j \end{aligned}$$

The difficulty is that we wish to find the value of $\boldsymbol{G}_h$ and $\boldsymbol{B}_h$ simultaneously.

3.1.2 Idea 2

$$\tilde{\boldsymbol{y}}_h = \boldsymbol{S}\check{\boldsymbol{G}}_h\check{\boldsymbol{B}}_h\hat{\boldsymbol{y}}_h,$$

where $\check{\boldsymbol{G}}_h$ is given using ols, wls, structural scaling, or shrinkage estimator of $\boldsymbol{W}_h$. $\check{\boldsymbol{B}}_h$ is an $n \times n$ diagonal matrix with elements of the main diagonal being either greater than or equal to zero. So $\check{\boldsymbol{B}}_h$ can adaptively changes given $\check{\boldsymbol{G}}_h$ and we can only estimate $\check{\boldsymbol{B}}_h$. But the reconciled forecasts are likely to no longer preserve the unbiasedness.

The optimization problem can be written as

$$\begin{aligned}\min_{\boldsymbol{G}_h} \left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\check{\boldsymbol{G}}_h\check{\boldsymbol{B}}_h\hat{\boldsymbol{y}}_h\right)' \boldsymbol{W}_h^{-1} \left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\check{\boldsymbol{G}}_h\check{\boldsymbol{B}}_h\hat{\boldsymbol{y}}_h\right) + \lambda ||\check{\boldsymbol{B}}_h\mathbf{1}||_1,\\ \text{s.t. } b_{ii} \geq 0 \text{ for } i = 1, \ldots, n\\ b_{ij} = 0 \text{ for } i \neq j\end{aligned}$$

R implementation

```r
library(CVXR)

y_hat <- c(10, 6, 5, 1, 4, 0, 2, 5)
S <- rbind(matrix(c(rep(1, 5), c(rep(1, 3), rep(0, 2)), c(rep(0, 3), rep(1, 2))),
                  nrow = 3, byrow = TRUE),
           diag(1, nrow = 5))

W_inv <- solve(diag(c(5,3,2,1,1,1,1,1))) # structural scaling approximation
G <- solve(t(S) %*% W_inv %*% S) %*% t(S) %*% W_inv

b <- Variable(8)
lambda <- 0.3

e <- y_hat - S %*% G  %*% diag(b) %*% y_hat
obj <- quad_form(e, W_inv) + lambda * norm1(b)
prob <- Problem(Minimize(obj), constraints = list(b >= 0))
res <- solve(prob)
res$getValue(b)
```

```
##                 [,1]
## [1,]  2.076187e+00
## [2,]  6.941964e-05
## [3,]  9.090869e-01
## [4,]  6.774912e-01
## [5,]  9.474978e-01
## [6,]  4.014566e-12
## [7,] -4.811095e-05
## [8,]  5.999817e-01
```

```
(B_check <- diag(round(as.vector(res$getValue(b)), digits = 3)))
```

```
##       [,1] [,2]  [,3]  [,4]  [,5] [,6] [,7] [,8]
## [1,] 2.076    0 0.000 0.000 0.000    0    0  0.0
## [2,] 0.000    0 0.000 0.000 0.000    0    0  0.0
## [3,] 0.000    0 0.909 0.000 0.000    0    0  0.0
## [4,] 0.000    0 0.000 0.677 0.000    0    0  0.0
## [5,] 0.000    0 0.000 0.000 0.947    0    0  0.0
## [6,] 0.000    0 0.000 0.000 0.000    0    0  0.0
## [7,] 0.000    0 0.000 0.000 0.000    0    0  0.0
## [8,] 0.000    0 0.000 0.000 0.000    0    0  0.6
```

```
(y_tilde <- as.vector(S %*% G %*% B_check %*% y_hat))
```

```
## [1] 10.923333  5.183500  5.739833  0.916500  4.027500  0.239500  1.369917
## [8]  4.369917
```

3.1.3 Idea 3

$$\tilde{\boldsymbol{y}}_h = \boldsymbol{S}\boldsymbol{G}_h\hat{\boldsymbol{y}}_h$$

According to Miyashiro & Takano (2015), we can use mixed logical programming and rewrite the problem as

$$\begin{aligned}
\min\nolimits_{\boldsymbol{G}_h} (\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}_h\hat{\boldsymbol{y}}_h)' \boldsymbol{W}_h^{-1} (\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}_h\hat{\boldsymbol{y}}_h) + \lambda \textstyle\sum_{j=1}^{n} b_j, \\
\text{s.t. } \boldsymbol{G}_h\boldsymbol{S} = \boldsymbol{I}_{n_b} \\
b_j = 0 \Rightarrow ||\boldsymbol{G}_{\cdot j}||_1 = 0 \quad (j = 1, 2, \ldots, n) \\
b_j \in \{0, 1\} \quad (j = 1, 2, \ldots, n)
\end{aligned}$$

3.2 Minimization problem

As already shown in Ben Taieb & Koo (2019),

$$\begin{aligned}
&\mathrm{E}\left[\|\boldsymbol{y}_{T+h} - \tilde{\boldsymbol{y}}_h\|_2^2 \mid \boldsymbol{I}_T\right] \\
&= \|\boldsymbol{S}\boldsymbol{G}\left(\mathrm{E}\left[\hat{\boldsymbol{y}}_h \mid \boldsymbol{I}_T\right] - \mathrm{E}\left[\boldsymbol{y}_{T+h} \mid \boldsymbol{I}_T\right]\right) + (\boldsymbol{S} - \boldsymbol{S}\boldsymbol{G}\boldsymbol{S})\mathrm{E}\left[\boldsymbol{b}_{T+h} \mid \boldsymbol{I}_T\right]\|_2^2 \\
&+ \mathrm{Tr}\left(\mathrm{Var}\left[\boldsymbol{y}_{T+h} - \tilde{\boldsymbol{y}}_h \mid \boldsymbol{I}_T\right]\right).
\end{aligned}$$

Under **the unbiasedness conditions (both for base and reconciled forecasts)**, minimizing the above loss reduces to the following problem:

$$\min_{\boldsymbol{G}_h \in \mathcal{G}} \mathrm{Tr}\left(\mathrm{Var}\left[\boldsymbol{y}_{T+h} - \tilde{\boldsymbol{y}}_h \mid \boldsymbol{I}_T\right]\right) \text{ s.t. } \boldsymbol{S}\boldsymbol{G}_{\boldsymbol{h}}\boldsymbol{S} = \boldsymbol{S}.$$

The trace minimization problem can be reformulated in terms of a linear equality constrained least squares problem as follows:

$$\min_{\boldsymbol{G}_h} \left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}_h\hat{\boldsymbol{y}}_h\right)' \boldsymbol{W}_h^{-1} \left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}_h\hat{\boldsymbol{y}}_h\right), \text{ s.t. } \boldsymbol{G}_h\boldsymbol{S} = \boldsymbol{I}_{n_b}$$

- The MinT optimization problem must impose **two unbiasedness conditions**.
- Such an assumption does not always hold in practice, especially when we aim to zero out some columns of $\boldsymbol{G}$.
- The out-of-sample forecasting accuracy of base forecasts is an important basis for determining which columns are zeroed out.

3.3 Relaxation of the unbiasedness assumptions

Ben Taieb & Koo (2019) consider the empirical risk minimization (ERM) problem:

$$\begin{aligned}
\hat{G}_{\text{ERMreg}} &= \underset{\boldsymbol{G} \in \mathcal{G}}{\operatorname{argmin}} \left\{\left\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}\boldsymbol{G}'\boldsymbol{S}'\right\|_F^2 / Nn + \lambda\|\operatorname{vec}(\boldsymbol{G})\|_1\right\}, \\
&= \underset{\boldsymbol{G} \in \mathcal{G}}{\operatorname{argmin}} \left\{\left\|\operatorname{vec}(\boldsymbol{Y}) - (\boldsymbol{S} \otimes \hat{\boldsymbol{Y}}) \operatorname{vec}\left(\boldsymbol{G}'\right)\right\|_2^2 / Nn + \lambda\|\operatorname{vec}(\boldsymbol{G})\|_1\right\}
\end{aligned}$$

The problem can be reduced to **a standard LASSO problem** with $\operatorname{vec}(\boldsymbol{Y})$ as dependent variable and $(\boldsymbol{S} \otimes \hat{\boldsymbol{Y}})$ as design matrix. it is a LASSO problem with $N \times n$ observations and $m \times N$ variables.

3.4 Ideas based on the assumption relaxation

3.4.1 Optimization problem

For our project. . .

We aim to deal with the minimization problem

$$
\begin{aligned}
\hat{G} =& \underset{\boldsymbol{G}\in\mathcal{G}}{\operatorname{argmin}} \left\{ \left\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}\boldsymbol{G}'\boldsymbol{S}'\right\|_F^2 /Nn + \lambda\sum_{j=1}^{n} z_j \right\}, \\
=& \underset{\boldsymbol{G}\in\mathcal{G}}{\operatorname{argmin}} \left\{ \left\|\operatorname{vec}(\boldsymbol{Y}) - (\boldsymbol{S}\otimes\hat{\boldsymbol{Y}}) \operatorname{vec}\left(\boldsymbol{G}'\right)\right\|_2^2 /Nn + \lambda\sum_{j=1}^{n} z_j \right\} \\
s.t. & -Mz_j \leq \sum_{i=0}^{m-1} |g_{j+in}| \leq Mz_j \text{ (Or quadratic form)} \\
& z_j \in \{0,1\} \quad \text{for } j = 1,\dots,n
\end{aligned}
$$

- Mixed integer programming (Do not have closed-form solution).
- The results are greatly influenced by the values of M and λ.
- If we assume all elements of $\boldsymbol{G}$ are non-negative, it can be solved by CVXR package even if this is a non-convex problem.

```
library(CVXR)


set.seed(123)
S <- rbind(matrix(c(rep(1, 5), c(rep(1, 3), rep(0, 2)), c(rep(0, 3), rep(1, 2))),
                  nrow = 3, byrow = TRUE),
           diag(1, nrow = 5))
B <- matrix(c(rnorm(10, 1, 1),
              rnorm(10, 4, 1),
              rnorm(10, 0, 1),
              rnorm(10, 2, 1),
              rnorm(10, 5, 1)),
            byrow = FALSE, nrow = 10)
Y <- B %*% t(S)
Y_hat <- matrix(c(rnorm(10, 10, 1),
                  rnorm(10, 6, 1),
                  rnorm(10, 5, 1),
                  rnorm(10, 1, 1),
                  rnorm(10, 4, 1),
                  rnorm(10, 0, 1),
                  rnorm(10, 2, 1),
                  rnorm(10, 5, 1)),
                byrow = FALSE, nrow = 10)

VecY <- as.vector(Y)
D <- kronecker(S, Y_hat)

b <- Variable(8, boolean = TRUE)
#G_trs <- Variable(8, 5)
GB_trs <- Variable(40)

lambda <- 0.1
M <- 100

loss <- sum_squares(VecY - D %*% GB_trs)/(10*8)
penalty <- lambda * norm1(b)
constraints <- list(
  GB_trs >= 0,
  sum_entries(GB_trs[(0:4)*8 + 1]) >= - M*b[1],
  sum_entries(GB_trs[(0:4)*8 + 2]) >= - M*b[2],
  sum_entries(GB_trs[(0:4)*8 + 3]) >= - M*b[3],
  sum_entries(GB_trs[(0:4)*8 + 4]) >= - M*b[4],
  sum_entries(GB_trs[(0:4)*8 + 5]) >= - M*b[5],
  sum_entries(GB_trs[(0:4)*8 + 6]) >= - M*b[6],
  sum_entries(GB_trs[(0:4)*8 + 7]) >= - M*b[7],
  sum_entries(GB_trs[(0:4)*8 + 8]) >= - M*b[8],
  sum_entries(GB_trs[(0:4)*8 + 1]) <=  M*b[1],
  sum_entries(GB_trs[(0:4)*8 + 2]) <=  M*b[2],
  sum_entries(GB_trs[(0:4)*8 + 3]) <=  M*b[3],
  sum_entries(GB_trs[(0:4)*8 + 4]) <=  M*b[4],
  sum_entries(GB_trs[(0:4)*8 + 5]) <=  M*b[5],
  sum_entries(GB_trs[(0:4)*8 + 6]) <=  M*b[6],
  sum_entries(GB_trs[(0:4)*8 + 7]) <=  M*b[7],
  sum_entries(GB_trs[(0:4)*8 + 8]) <=  M*b[8]
)

prob <- Problem(Minimize(loss + penalty), constraints)
res <- solve(prob)
(res$getValue(b))
```

```
##      [,1]
## [1,]    0
## [2,]    0
## [3,]    1
## [4,]    0
## [5,]    0
## [6,]    1
## [7,]    1
## [8,]    1
```

```
t(matrix(round(res$getValue(GB_trs), digits = 5), 8, 5, byrow = FALSE))
```

```
##      [,1] [,2]    [,3] [,4] [,5]    [,6]    [,7]    [,8]
## [1,]    0    0 0.21966    0    0 0.26497 0.06571 0.00000
## [2,]    0    0 0.65243    0    0 0.37896 0.72396 0.00000
## [3,]    0    0 0.00000    0    0 0.41389 0.00000 0.00000
## [4,]    0    0 0.26735    0    0 0.00000 0.21742 0.14529
## [5,]    0    0 0.14430    0    0 0.10789 0.52331 0.69977
```

3.4.2 Possible benchmark based on iterative process

$$
\begin{aligned}
\hat{G} =& \operatorname*{argmin}_{\boldsymbol{G}\in\mathcal{G}} \left\{ \left\| \boldsymbol{Y} - \hat{\boldsymbol{Y}} \boldsymbol{B}'\boldsymbol{G}'\boldsymbol{S}' \right\|_F^2 /Nn + \lambda \|B\|_1 \right\}, \\
=& \operatorname*{argmin}_{\boldsymbol{G}\in\mathcal{G}} \left\{ \left\| \operatorname{vec}(\boldsymbol{Y}) - (\boldsymbol{S} \otimes \hat{\boldsymbol{Y}}) \operatorname{vec}\left(\boldsymbol{B}'\boldsymbol{G}'\right) \right\|_2^2 /Nn + \lambda \|B\|_1 \right\} \\
=& \operatorname*{argmin}_{\boldsymbol{G}\in\mathcal{G}} \left\{ \left\| \operatorname{vec}(\boldsymbol{Y}) - (\boldsymbol{S} \otimes \hat{\boldsymbol{Y}})(\boldsymbol{G} \otimes \boldsymbol{I}_n) \operatorname{vec}\left(\boldsymbol{B}'\right) \right\|_2^2 /Nn + \lambda \|B\|_1 \right\} \\
=& \operatorname*{argmin}_{\boldsymbol{G}\in\mathcal{G}} \left\{ \left\| \operatorname{vec}(\boldsymbol{Y}) - (\boldsymbol{S} \otimes \hat{\boldsymbol{Y}})(\boldsymbol{I}_n \otimes \boldsymbol{B}') \operatorname{vec}\left(\boldsymbol{G}'\right) \right\|_2^2 /Nn + \lambda \|B\|_1 \right\} \\
s.t. \quad & b_{ii} \in \{0,1\} \quad \text{for } i = 1, \dots, n \\
& b_{ij} = 0 \quad \text{for } i \neq j
\end{aligned}
$$

- Initial $\boldsymbol{G}^0 \Rightarrow \boldsymbol{B}^1$ (Lasso problem) $\Rightarrow \boldsymbol{G}^1$ (Analytical solution) $\Rightarrow \dots$

3.5 ROI introduction

The R Optimization Infrastructure (ROI) package is an R **interface** which provides an extensible infrastructure to model linear, quadratic, conic and general nonlinear optimization problems in a consistent way.

ROI provides the modeling capabilities and manages the **plugins**, the plugins add the solvers to ROI.

Optimization problem

$$\begin{aligned} \text{minimize } & f_0(x) \\ \text{s.t. } & f_i(x) \le b_i, \quad i = 1, \dots, m \end{aligned} \tag{1}$$

1. Linear programming (LP)

   All f_i $(i = 0, \dots, m)$ in Equation 1 are linear.

   All LPs are convex.

2. Quadratic programming (QP)

   The objective function f_0 contains a quadratic part in addition to the linear term. QPs can be expressed in the following manner:

   $$\begin{aligned} \text{minimize } & \frac{1}{2} x^\top Q_0 x + a_0^\top x \\ \text{s.t. } & Ax \le b \end{aligned}$$

   A QP is convex if and only if Q_0 is positive semidefinite.

   A generalization of the QP is the quadratically constrained quadratic program (QCQP):

   $$\begin{aligned} \text{minimize } & \frac{1}{2} x^\top Q_0 x + a_0^\top x \\ \text{s.t. } & \frac{1}{2} x^\top Q_i x + a_i^\top x \le b_i, \quad i = 1, \dots, m \end{aligned}$$

   A QCQP is convex if and only if all $Q_i (i = 0, \dots, m)$ are positive semidefinite.

3. Conic programming (CP)

   CP refers to a class of problems designed to model convex OPs. A CP is commonly defined as:

   $$\begin{aligned} \text{minimize } & a_0^\top x \\ \text{s.t. } & Ax + s = b \\ & s \in \mathcal{K} \end{aligned}$$

where s is a slack variable that is added to an inequality constraint to transform it to an equality, and the set $\mathcal{K}$ is a nonempty closed convex cone.

Nonlinear objective functions are expressed in epigraph form:

$$\begin{aligned} &\text{minimize } t \\ &\quad \text{s.t. } f_0(x) \leq t \\ &\qquad\quad f_i(x) \leq b_i \end{aligned}$$

- Zero cone and free cone

  $$\mathcal{K}_{\text{zero }} = \{0\}, \mathcal{K}_{\text{free }} = \mathbb{R} = \mathcal{K}_{\text{zero }}^* .$$

  E.g., $s_i \in \mathcal{K}_{\text{zero }} \iff s_i = b_i - a_i^\top x = 0 \iff a_i^\top x = b_i$.

- Linear cone (non-negative orthant)

  $$\mathcal{K}_{\text{lin}} = \{x \in \mathbb{R} \mid x \geq 0\}.$$

  E.g., $s_i \in \mathcal{K}_{\text{lin }} \iff s_i = b_i - a_i^\top x \geq 0 \iff a_i^\top x \leq b_i$.

- Second-order cone

  $$\mathcal{K}_{\text{soc}}^n = \left\{(t, x) \in \mathbb{R}^n \mid x \in \mathbb{R}^{n-1}, t \in \mathbb{R}, \|x\|_2 \leq t\right\}.$$

  E.g., $s_i \in \mathcal{K}_{\text{soc}}^n \iff s_i = b_i - a_i^\top x \geq 0 \iff a_i^\top x \leq b_i$.

- ...

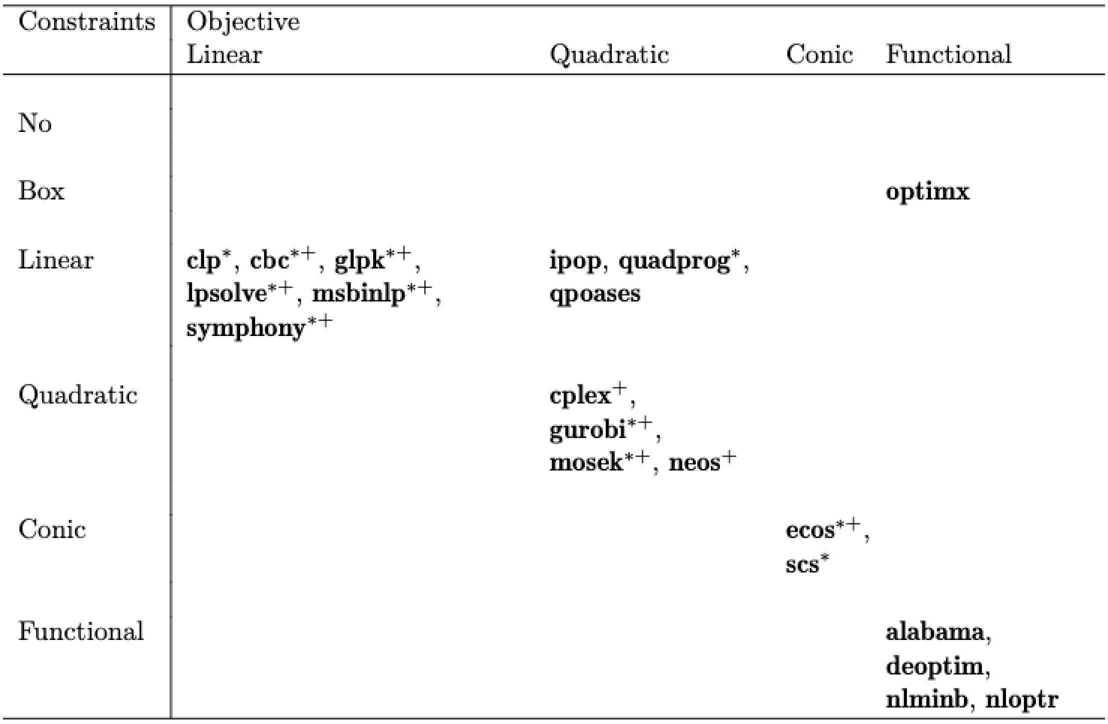
4. Nonlinear programming (NLP)

   At least one f_i, $i = 0, \ldots, m$ in Equation 1 is not linear.

   NLPs are not required to be convex, which makes it in general hard to obtain a reliable global solution.

5. Mixed integer programming (MIP)

   Additional constraints: some of the objective variables can only take integer values.



Constraints	Objective Linear	Quadratic	Conic	Functional
No				
Box				**optimx**
Linear	**clp***, **cbc**$^{*+}$, **glpk**$^{*+}$, **lpsolve**$^{*+}$, **msbinlp**$^{*+}$, **symphony**$^{*+}$	**ipop**, **quadprog***, **qpoases**		
Quadratic		**cplex**$^+$, **gurobi**$^{*+}$, **mosek**$^{*+}$, **neos**$^+$		
Conic			**ecos**$^{*+}$, **scs***	
Functional				**alabama**, **deoptim**, **nlminb**, **nloptr**

Table 4: Currently available **ROI** plug-ins displayed based on the types of optimization problems they are applicable to. Here * indicates that the solver is restricted to convex problems and $^+$ indicates that the solver can model integer constraints. Note all the plug-ins have the prefix **ROI.plugin** and the modeling capabilities of the plug-ins do not necessarily represent the modeling capabilities of the underlying solvers.

3.6 Optimization problem solving

3.6.1 Rewrite the optimization problem

Original problem:

$$\begin{aligned}
&\underset{\boldsymbol{G},\boldsymbol{z}}{\operatorname{argmin}}\left\{\left\|\boldsymbol{Y}-\hat{\boldsymbol{Y}}\boldsymbol{G}'\boldsymbol{S}'\right\|_F^2/Nn+\lambda\sum_{j=1}^{n}z_j\right\},\\
=&\underset{\boldsymbol{G},\boldsymbol{z}}{\operatorname{argmin}}\left\{\left\|\operatorname{vec}(\boldsymbol{Y})-(\boldsymbol{S}\otimes\hat{\boldsymbol{Y}})\operatorname{vec}\left(\boldsymbol{G}'\right)\right\|_2^2/Nn+\lambda\sum_{j=1}^{n}z_j\right\}\\
&s.t.-Mz_j\leq\sum_{i=0}^{m-1}|g_{j+in}|\leq Mz_j\ \text{(Or quadratic form)}\\
&z_j\in\{0,1\}\quad\text{for } j=1,\dots,n
\end{aligned}\tag{2}$$

Rewrite the problem:

$$
\begin{aligned}
&\underset{\boldsymbol{G},\boldsymbol{z}}{\operatorname{argmin}} \left\{ \left\| \operatorname{vec}(\boldsymbol{Y}) - (\boldsymbol{S} \otimes \hat{\boldsymbol{Y}}) \operatorname{vec}(\boldsymbol{G}') \right\|_2^2 / Nn + \lambda \sum_{j=1}^{n} z_j \right\} \\
=&\underset{\boldsymbol{g},\boldsymbol{z}}{\operatorname{argmin}} \left\{ \| \boldsymbol{y} - \boldsymbol{X}\boldsymbol{g} \|_2^2 / Nn + \lambda \boldsymbol{1}'\boldsymbol{z} \right\} \\
=&\underset{\boldsymbol{g},\boldsymbol{\gamma},\boldsymbol{z}}{\operatorname{argmin}} \left\{ \frac{1}{Nn} \boldsymbol{\gamma}'\boldsymbol{\gamma} + \lambda \boldsymbol{1}'\boldsymbol{z} \right\} \\
\text{s.t. } & \boldsymbol{y} - \boldsymbol{X}\boldsymbol{g} = \boldsymbol{\gamma} \\
& \sum_{i=0}^{m-1} (g_{j+in})^2 \leq M z_j \quad \text{for } j = 1, \dots, n \\
& \boldsymbol{z} \in \{0, 1\}^n
\end{aligned}
\tag{3}
$$

3.6.2 ROI code logic

QP with LC and QC:

$$
\begin{aligned}
\underset{\beta = (\boldsymbol{g}'_{mn}, \boldsymbol{\gamma}'_{Nn}, \boldsymbol{z}'_n)'}{\operatorname{argmin}} \quad & \frac{1}{2} \boldsymbol{\beta}' \boldsymbol{Q}_0 \boldsymbol{\beta} + \boldsymbol{a}'_0 \boldsymbol{\beta} \\
\text{s.t. } & \boldsymbol{A}\boldsymbol{\beta} = \boldsymbol{y} \\
& \frac{1}{2} \boldsymbol{\beta}' \boldsymbol{Q}_j \boldsymbol{\beta} + \boldsymbol{a}'_j \boldsymbol{\beta} \leq b_j, \quad j = 1, \dots, n \\
& \boldsymbol{z} \in \{0, 1\}^n
\end{aligned}
$$

- $\boldsymbol{Q}_0 = \begin{pmatrix} \boldsymbol{O}_{mn} & & \\ & \frac{2}{Nn} \boldsymbol{1}_{Nn} & \\ & & \boldsymbol{O}_n \end{pmatrix}$ and $\boldsymbol{a}_0 = (\boldsymbol{0}'_{mn+Nn}, \lambda \boldsymbol{1}'_n)'$.
- $\boldsymbol{A} = (\boldsymbol{X}_{Nn \times mn} | \boldsymbol{1}_{Nn} | \boldsymbol{O}_{Nn \times n})$.
- For $j = 1, \dots, n$,
  - $\boldsymbol{Q}_j = \begin{pmatrix} 2\boldsymbol{D}_{mn} & \\ & \boldsymbol{O}_{Nn+n} \end{pmatrix}$ with $d_{j+i \times n, j+i \times n} = 1$ $(i = 0, \dots, m-1)$ and other elements are zeros.
  - $\boldsymbol{a}_j = (\boldsymbol{0}'_{mn+Nn}, -M\boldsymbol{d}'_n)'$ with $d_j = 1$ and other elements are zeros.

3.6.3 ROI implementation

```
Sys.setenv(ROI_LOAD_PLUGINS = "FALSE")
library(ROI)
library(slam)
```

```r
library(magrittr)
library(ROI.plugin.neos)
library(ROI.plugin.gurobi)

dbind <- function(...) {
  ## sparse matrices construction
  .dbind <- function(x, y) {
    A <- simple_triplet_zero_matrix(NROW(x), NCOL(y))
    B <- simple_triplet_zero_matrix(NROW(y), NCOL(x))
    rbind(cbind(x, A), cbind(B, y))
  }
  Reduce(.dbind, list(...))
}

# Example data
set.seed(123)
S <- rbind(matrix(c(rep(1, 5), c(rep(1, 3), rep(0, 2)), c(rep(0, 3), rep(1, 2))),
                  nrow = 3, byrow = TRUE),
           diag(1, nrow = 5))
B <- matrix(c(rnorm(10, 1, 1),
              rnorm(10, 4, 1),
              rnorm(10, 0, 1),
              rnorm(10, 2, 1),
              rnorm(10, 5, 1)),
            byrow = FALSE, nrow = 10)
Y <- B %*% t(S)
Y_hat <- matrix(c(rnorm(10, 10, 1),
                  rnorm(10, 6, 1),
                  rnorm(10, 5, 1),
                  rnorm(10, 1, 1),
                  rnorm(10, 4, 1),
                  rnorm(10, 0, 1),
                  rnorm(10, 2, 1),
                  rnorm(10, 5, 1)),
                byrow = FALSE, nrow = 10)
```

```r
VecY <- as.vector(Y)
D <- kronecker(S, Y_hat)

# Quadratic optimization problem
gp_op <- function(x, y, n, m, lambda, M) {
  ## x: kronecker(S, Y_hat)
  ## y: vec(Y)
  ## lambda: Lagrange multiplier
  ## n: number of all series
  ## m: number of bottom-level series
  ## M: big-M
  stzm <- simple_triplet_zero_matrix
  stdm <- simple_triplet_diag_matrix

  Nn <- NROW(x); mn <- NCOL(x)
  Q0 <- dbind(stzm(mn), stdm(2/Nn, Nn), stzm(n))
  a0 <- c(g = double(mn), ga = double(Nn), z = lambda * rep(1, n))
  op <- OP(objective = Q_objective(Q = Q0, L = a0))

  ## y - X %*% g = gamma  <=>  X %*% g + gamma = y
  A1 <- cbind(x, stdm(1, Nn), stzm(Nn, n))
  LC1 <- L_constraint(A1, eq(Nn), y)
  ## \sum_{i=0}^{m-1} (g_{j+in})^2 - M z_j <= 0 for j = 1,...,n
  QNULL <- diag(0, mn + Nn + n)
  LNULL <- c(double(mn), double(Nn), double(n))
  Q <- lapply(1:n, function(j){
    i <- 0:(m - 1)
    Q_j <- QNULL
    diag(Q_j)[j + i*n] <- 2
    Q_j
  })
  L <- sapply(1:n, function(j){
    L_j <- LNULL
    L_j[mn + Nn + j] <- -M
    L_j
  }) %>% t()
  QC1 <- Q_constraint(Q = Q,
                      L = L,
                      dir = rep("<=", n),
                      rhs = rep(0, n))

  constraints(op) <- rbind(LC1, QC1)
  bounds(op) <- V_bound(li = 1:(mn + Nn), lb = rep.int(-Inf, mn + Nn), nobj = mn + Nn + n)
  types(op) <- c(rep("C", mn + Nn), rep("B", n))
  op
}
op <- gp_op(x = D, y = VecY, n = NROW(S), m = NCOL(S), lambda = 0.1, M = 100)

# Optimal solution - solver = "neos"
job_neos <- ROI_solve(op, "neos", email = "xiaoqian.wang@monash.edu")
## str(job_neos)
slt_neos <- solution(job_neos)
(z <- tail(slt_neos, NROW(S)))
```

```
## [1] 1 1 1 1 1 1 0 1
```

```
(G_neos <- matrix(slt_neos[1:(NCOL(S)*NROW(S))],
                  nrow = NROW(S), ncol = NCOL(S), byrow = FALSE) %>%
  t() %>%
  round(digits = 3))
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6] [,7]   [,8]
## [1,]  0.053 -0.873  0.976  0.389  0.548  1.065    0 -0.238
## [2,]  0.301 -0.293  0.593 -0.298  0.807  1.278    0 -0.512
## [3,] -0.348  0.340 -0.268  0.938 -0.090 -0.046    0  0.295
## [4,]  0.132 -0.501  0.459  0.281  0.219  0.199    0  0.132
## [5,]  0.436 -0.049 -0.351  0.147 -0.356 -0.112    0  0.776
```

```
# Optimal solution - solver = "gurobi"
job_gurobi <- ROI_solve(op, "gurobi")
 ## Register a Gurobi account as an academic user, request for a license, and download the cur
## str(job_gurobi)
slt_gurobi <- solution(job_gurobi)
(z <- tail(slt_gurobi, NROW(S)))
```

```
## [1] 1 1 1 1 1 1 0 1
```

```
(G_gurobi <- matrix(slt_gurobi[1:(NCOL(S)*NROW(S))],
                    nrow = NROW(S), ncol = NCOL(S), byrow = FALSE) %>%
    t() %>%
    round(digits = 3))
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6] [,7]   [,8]
## [1,]  0.053 -0.873  0.976  0.389  0.548  1.065    0 -0.238
## [2,]  0.301 -0.293  0.593 -0.298  0.807  1.278    0 -0.512
## [3,] -0.348  0.340 -0.268  0.938 -0.090 -0.046    0  0.295
## [4,]  0.132 -0.501  0.459  0.281  0.219  0.199    0  0.132
## [5,]  0.436 -0.049 -0.351  0.147 -0.356 -0.112    0  0.776
```

3.7 Hyperparameters

3.7.1 Big-M

3.7.2 λ

We can use cross-validation to select the best value of λ, as implemented in the glmnet package in R.

The glmnet package computes the solutions for **a decreasing sequence of values for** λ, starting at the smallest value $\lambda_{\max}$ for which the entire vector $\hat{\beta} = 0$. And then it selects a minimum value $\lambda_{\min} = \epsilon\lambda_{\max}$, and construct a sequence of K values of λ decreasing from $\lambda_{\max}$ to $\lambda_{\min}$ on the log scale. Typical values are $\epsilon = 0.001$ and $K = 100$.

4 Subset selection with shrinkage

4.1 Subset selection + shrinkage

> Mazumder, R., Radchenko, P., & Dedieu, A. (2022). Subset selection with shrinkage: Sparse linear modeling when the SNR is low. Operations Research.

Best subset selection problem

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad \|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_0 \leq k$$

1. Computationally infeasible

2. $\hat{\boldsymbol{\beta}}$ is instable, especially when SNR (signal to noise ratio) is low, the pairwise (sample) correlations among the features are high, and n is relatively small compared to p. **Overfit**

Subset selection with shrinkage

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \underbrace{\lambda\|\boldsymbol{\beta}\|_q}_{\text{Shrinkage}} \quad \text{s.t.} \underbrace{\|\boldsymbol{\beta}\|_0 \leq k}_{\text{Sparsity}}.$$

- Sparsity & Shrinkage

MIO formulations for this problem:

$$\begin{gathered}
\text{minimize } u/2+\lambda v \\
\text{s.t. } \|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 \leq u \\
\|\boldsymbol{\beta}\|_q \leq v \\
-\mathcal{M} z_j \leq \beta_j \leq \mathcal{M} z_j, j \in [p]; \\
\mathbf{z} \in \{0,1\}^p; \\
\textstyle\sum_j z_j = k
\end{gathered}$$

- Algorithms are written in Python. The MIO formulation is solved with Gurobi's mixed integer programming solver.

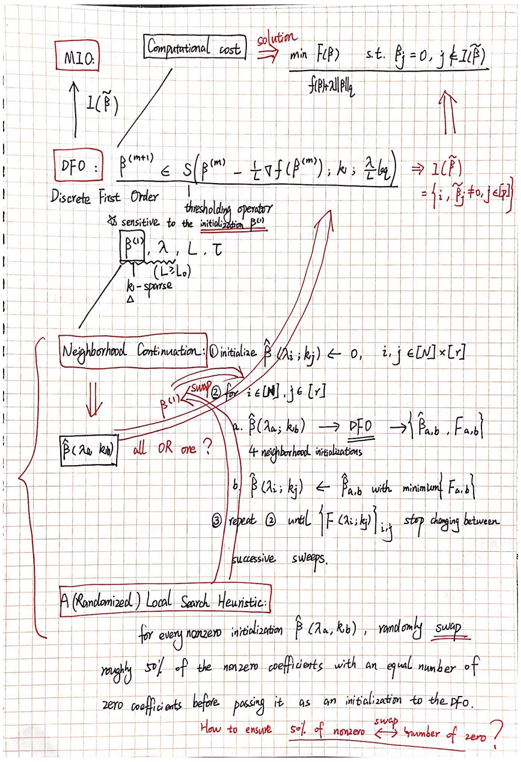
4.2 L_0L_q-regularized regression problem

Hazimeh, H., & Mazumder, R. (2020). Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. Operations Research, 68(5), 1517-1537.

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \frac{1}{2}\|y-X\beta\|_2^2+\lambda_0\|\beta\|_0+\lambda_1\|\beta\|_1+\lambda_2\|\beta\|_2^2$$

An overview of the different classes of local minima and establish the following hierarchy:

$$\begin{array}{ccccccccc}
\text{FSI}(k) & \subseteq & \text{PSI}(k) & \subseteq & \text{CW} & \subseteq & \text{IHT} & \subseteq & \text{Stationary} \\
\text{Minima} & & \text{Minima} & & \text{Minima} & & \text{Minima} & & \text{Solutions}
\end{array}$$


MIO:
Computational cost
solution
$\Rightarrow \min F(\beta)$ s.t. $\beta_j = 0, j \notin I(\tilde{\beta})$
$f(\beta)+\lambda\|\beta\|_q$
$I(\tilde{\beta})$
DFO :
$\beta^{(m+1)} \in S(\beta^{(m)} - \frac{1}{L}\nabla f(\beta^{(m)}); k; \frac{\lambda}{L}b_q)$
$\Rightarrow I(\tilde{\beta}) = \{i, \tilde{\beta}_j \neq 0, j \in [p]\}$
Discrete First Order
thresholding operator
sensitive to the initialization $\beta^{(1)}$
$\beta^{(1)}, \lambda, L, \tau$
$(L \geq L_0)$
k_1-sparse
Neighborhood Continuation:
① initialize $\hat{\beta}(\lambda_i; k_j) \leftarrow 0$, $i, j \in [N] \times [r]$
swap
$\beta^{(1)}$
② for $i \in [N], j \in [r]$
a. $\hat{\beta}(\lambda_a; k_b) \rightarrow$ DFO $\rightarrow \{\hat{\beta}_{a,b}, F_{a,b}\}$
4 neighborhood initializations
$\hat{\beta}(\lambda_a\ k_b)$
all OR one ?
b. $\hat{\beta}(\lambda_i; k_j) \leftarrow \hat{\beta}_{a,b}$ with minimum $\{F_{a,b}\}$
③ repeat ② until $\{F(\lambda_i; k_j)\}_{i,j}$ stop changing between successive sweeps.
A (Randomized) Local Search Heuristic:

for every nonzero initialization $\hat{\beta}(\lambda_a, k_b)$, randomly swap roughly 50% of the nonzero coefficients with an equal number of zero coefficients before passing it as an initialization to the DFO.

How to ensure 50% of nonzero $\overset{swap}{\longleftrightarrow}$ number of zero ?

Figure 2: Mazumder et al. (2022) framework

- For sufficiently large k, FSI(k) and PSI(k) minima coincide with the class of global minimizers.

1. Cyclic coordinate descent
    - full minimization in every coordinate: β^{k+1} is obtained by updating the ith coordinate (with others held fixed).
    - converges to CW minima
2. Cyclic coordinate descent & local combinatorial optimization algorithms (iterative)
    - PSI(k)
    - FSI(k)

- It cannot provide certificates of optimality.
- L0Learn: an extensible C++ toolkit with an R interface

4.3 Group l_0 problem

Hazimeh, H., Mazumder, R., & Radchenko, P. (2021). Grouped variable selection with discrete optimization: Computational and statistical perspectives. arXiv preprint

Suppose that the p predictors are divided into q pre-specified, non-overlapping groups.

Group l_0 with ridge regularization

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_0 \sum_{g=1}^{q} \mathbf{1}\left(\boldsymbol{\beta}_g \neq \mathbf{0}\right) + \lambda_2 \|\boldsymbol{\beta}\|_2^2$$

1. Approximate algorithms (local minimizers)

- cyclic block coordinate descent (BCD) ⟶ (improved by) local combinatorial search
- do not deliver certificates of optimality

2. MIP formulation

$$\begin{aligned} \underset{\boldsymbol{\theta},\mathbf{z},\boldsymbol{s}}{\text{minimize}} & \tilde{\ell}(\boldsymbol{\theta}) + \lambda_0 \sum_{g=1}^{q} z_g + \lambda_1 \sum_{g=1}^{q} \|\boldsymbol{\theta}_g\|_2 + \lambda_2 \sum_{g=1}^{q} s_g, \\ \text{s.t. } & \|\boldsymbol{\theta}_g\|_2 \leq \mathcal{M}_{\mathrm{U}} z_g, g \in [q] \\ & \|\boldsymbol{\theta}_g\|_2^2 \leq s_g z_g, \quad g \in [q] \\ & z_g \in \{0,1\}, s_g \geq 0, \quad g \in [q]. \end{aligned}$$

- Propose a specialized, nonlinear Branch-and-Bound (BnB) framework.
- Subproblem solver: active-set algorithm, which exploits sparsity by considering a reduced problem restricted to a small subset of groups.
- Upper bounds: obtain a new upper bound by restricting optimization, leading to aggressive pruning in the search tree, which can reduce the overall runtime.
- Branching and search strategies:
  - for branching, use maximum fractional branching.
  - for search, use breadth-first search and switch to depth-first search if memory issues are encountered.
- Solve the associated MIP problem to certified optimality.
- L0Group written in Python.

4.4 Subset selection with shrinkage under unbiasedness assumptions

The vectorization is frequently used together with the Kronecker product to express matrix multiplication as a linear transformation on matrices.

$$\mathrm{Vec}(ABC) = (C' \otimes A)\,\mathrm{vec}(B)$$

With two unbiasedness conditions, the trace minimization (MinT) problem can be reformulated in terms of a linear equality constrained least squares problem as follow.

$$\min_{\boldsymbol{G}} \frac{1}{2}\left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}\hat{\boldsymbol{y}}_h\right)' \boldsymbol{W}_h^{-1}\left(\hat{\boldsymbol{y}}_h - \boldsymbol{S}\boldsymbol{G}\hat{\boldsymbol{y}}_h\right) \quad \text{s.t. } \boldsymbol{G}\boldsymbol{S} = \boldsymbol{I}_{nb}$$

$$\Downarrow$$

$$\min_{\boldsymbol{G}} \frac{1}{2}\left(\hat{\boldsymbol{y}}_h - \left(\hat{\boldsymbol{y}}_h' \otimes \boldsymbol{S}\right)\mathrm{vec}(\boldsymbol{G})\right)' \boldsymbol{W}_h^{-1}\left(\hat{\boldsymbol{y}}_h - \left(\hat{\boldsymbol{y}}_h \otimes \boldsymbol{S}\right)\mathrm{vec}(\boldsymbol{G})\right) \quad \text{s.t. } \boldsymbol{G}\boldsymbol{S} = \boldsymbol{I}_{nb}$$

If we consider subset selection with shrinkage which also imposes the two unbiasedness conditions of MinT, we have

$$\begin{aligned}
\min_{\boldsymbol{G}} &\frac{1}{2}\left(\hat{\boldsymbol{y}}_h - \left(\hat{\boldsymbol{y}}_h' \otimes \boldsymbol{S}\right)\mathrm{vec}(\boldsymbol{G})\right)' \boldsymbol{W}_h^{-1}\left(\hat{\boldsymbol{y}}_h - \left(\hat{\boldsymbol{y}}_h' \otimes \boldsymbol{S}\right)\mathrm{vec}(\boldsymbol{G})\right) \\
&\quad + \lambda_0 \sum_{j=1}^{n} \left\|\boldsymbol{G}_{\cdot j}\right\|_0 + \lambda_1 \left\|\mathrm{vec}\left(\boldsymbol{G} - \boldsymbol{G}^0\right)\right\|_1 + \lambda_2 \left\|\mathrm{vec}\left(\boldsymbol{G} - \boldsymbol{G}^0\right)\right\|_2^2 \\
\text{s.t. } &\boldsymbol{G}\boldsymbol{S} = \boldsymbol{I}_{nb}
\end{aligned}$$

where $\boldsymbol{G}^0$ can be a benchmark weight matrix estimated by MinT or other methods, such as bottom-up and top-down.

$$\begin{aligned}
\min_{\mathrm{vec}(\boldsymbol{G}),\boldsymbol{z},\check{\boldsymbol{e}},\boldsymbol{d}^+,\boldsymbol{g}^+} \; & \frac{1}{2}\check{\boldsymbol{e}}'\boldsymbol{W}_h^{-1}\check{\boldsymbol{e}} + \lambda_0 \sum_{j=1}^{n} z_j + \lambda_1 \boldsymbol{1}'\boldsymbol{d}^+ + \lambda_2 \boldsymbol{d}^{+\prime}\boldsymbol{d}^+ \\
\text{s.t. } & \hat{\boldsymbol{y}}_h - \left(\hat{\boldsymbol{y}}_h' \otimes \boldsymbol{S}\right)\mathrm{vec}(\boldsymbol{G}) = \check{\boldsymbol{e}} \quad \ldots (C1) \\
& \boldsymbol{G}\boldsymbol{S} = \boldsymbol{I}_{n_b} \Leftrightarrow \left(\boldsymbol{S}' \otimes \boldsymbol{I}_{n_b}\right)\mathrm{vec}(\boldsymbol{G}) = \mathrm{vec}\left(\boldsymbol{I}_{n_b}\right) \quad \ldots (C2) \\
& \sum_{i=1}^{n_b} \boldsymbol{g}^+_{i+(j-1)n_b} \leqslant M z_j, \quad j \in [n] \quad \ldots (C3) \\
& \boldsymbol{g}^+ \geqslant \mathrm{vec}(\boldsymbol{G}) \quad \ldots (C4) \\
& \boldsymbol{g}^+ \geqslant -\mathrm{vec}(\boldsymbol{G}) \quad \ldots (C5) \\
& \boldsymbol{d}^+ \geqslant \mathrm{vec}\left(\boldsymbol{G} - \boldsymbol{G}^0\right) \quad \ldots (C6) \\
& \boldsymbol{d}^+ \geqslant -\mathrm{vec}\left(\boldsymbol{G} - \boldsymbol{G}^0\right) \quad \ldots (C7) \\
& z_j \in \{0, 1\}, \quad j \in [n]
\end{aligned}$$

5 Other issues

5.1 Temporal reconciliation

- Remove all nodes in a level or several levels.

5.2 Non-negative constraints