

# Subset Selection for Forecast Reconciliation

Xiaoqian Wang

2023-04-26

## 1 Hierarchical time series

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

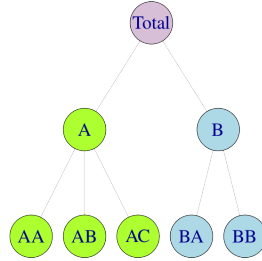


Figure 1: A 2-level hierarchical tree structure

For example,

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

## 2 Linear forecast reconciliation

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

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

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

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

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

### 3 Subset selection

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

*How to mathematically describe the constraint?*

#### 3.1 Ideas based on MinT

##### 3.1.1 Idea 1

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

where  $\mathbf{B}_h$  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

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

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

### 3.1.2 Idea 2

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

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

The optimization problem can be written as

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

### R implementation

```
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

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

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

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

### 3.2 Minimization problem

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

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

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

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

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

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

- 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  $\mathbf{G}$ .
- 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:

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

The problem can be reduced to a **standard LASSO problem** with  $\text{vec}(\mathbf{Y})$  as dependent variable and  $(\mathbf{S} \otimes \hat{\mathbf{Y}})$  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

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

- Mixed integer programming (Do not have closed-form solution).
- The results are greatly influenced by the values of  $M$  and  $\lambda$ .
- If we assume all elements of  $G$  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_trsr[(0:4)*8 + 7]) <= M*b[7],
sum_entries(GB_trsr[(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_trsr), 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

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



- Initial  $\mathbf{G}^0 \Rightarrow \mathbf{B}^1$  (Lasso problem)  $\Rightarrow \mathbf{G}^1$  (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

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

##### 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:

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

**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):

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

**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:

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

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:

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

- Zero cone and free cone

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

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

- Linear cone (non-negative orthant)

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

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

- Second-order cone

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

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

- ...

#### 4. Nonlinear programming (NLP)

At least one  $f_i$ ,  $i = 0, \dots, m$  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				<b>optimx</b>
Linear	<b>clp*</b> , <b>cbc**</b> , <b>glpk**</b> , <b>lpsolve**</b> , <b>msbinlp**</b> , <b>symphony**</b>	<b>ipop</b> , <b>quadprog*</b> , <b>qpoases</b>		
Quadratic		<b>cplex*</b> , <b>gurobi**</b> , <b>mosek**</b> , <b>neos*</b>		
Conic			<b>ecos**</b> , <b>scs*</b>	
Functional				<b>alabama</b> , <b>deoptim</b> , <b>nlminb</b> , <b>nloptr</b>

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:

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

Rewrite the problem:

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

### 3.6.2 ROI code logic

QP with LC and QC:

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

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

### 3.6.3 ROI implementation

```

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

```

```

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)

```

```

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))

  ##  $y - X \%*\% g = \text{gamma} \iff X \%*\% g + \text{gamma} = y$ 
  A1 <- cbind(x, stdm(1, Nn), stzm(Nn, n))
  LC1 <- L_constraint(A1, eq(Nn), y)
  ##  $\sum_{i=0}^{m-1} (g_{j+in})^2 - M z_j \leq 0$  for  $j = 1, \dots, n$ 
  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)))
# (G_neos <- matrix(slt_neos[1:(NCOL(S)*NROW(S))],
#                   nrow = NROW(S), ncol = NCOL(S), byrow = FALSE) %>%
#   t() %>%
#   round(digits = 3))

# 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 $\lambda$

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

The glmnet package computes the solutions for **a decreasing sequence of values for**  $\lambda$ , 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  $\lambda$  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

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

1. Computationally infeasible
2.  $\hat{\beta}$  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

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



- Sparsity & Shrinkage

MIO formulations for this problem:

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

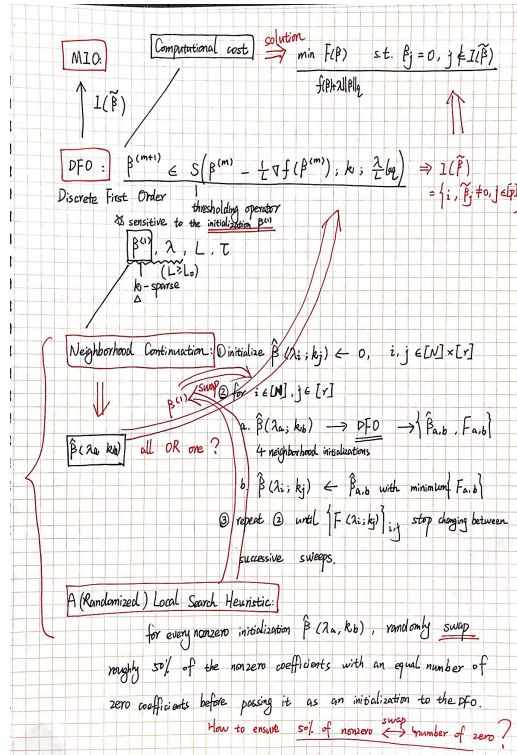


Figure 2: Mazumder et al. (2022) framework

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

## 4.2 $L_0 L_q$ -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.

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

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

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

- 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:  $\beta^{k+1}$  is obtained by updating the  $i$ th 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

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

#### 1. Approximate algorithms (local minimizers)

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

#### 2. MIP formulation

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

- 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.

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

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

$$\begin{aligned}
& \min_{\mathbf{G}} \frac{1}{2} (\hat{\mathbf{y}}_h - \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_h)' \mathbf{W}_h^{-1} (\hat{\mathbf{y}}_h - \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_h) \quad \text{s.t. } \mathbf{G}\mathbf{S} = \mathbf{I}_{n_b} \\
& \quad \Downarrow \\
& \min_{\mathbf{G}} \frac{1}{2} (\hat{\mathbf{y}}_h - (\hat{\mathbf{y}}_h' \otimes \mathbf{S}) \text{vec}(\mathbf{G}))' \mathbf{W}_h^{-1} (\hat{\mathbf{y}}_h - (\hat{\mathbf{y}}_h' \otimes \mathbf{S}) \text{vec}(\mathbf{G})) \quad \text{s.t. } \mathbf{G}\mathbf{S} = \mathbf{I}_{n_b}
\end{aligned}$$

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

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

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

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

#### 4.4.1 R implementation

Hierarchical structure & simple example

```

# Hierarchical structure
S <- rbind(rep(1, 4),
           c(1, 1, 0, 0), c(0, 0, 1, 1),
           diag(1, 4))

# OLS and WLSs estimator
W_I <- diag(7)

W_s <- S %%% matrix(1, nrow = NCOL(S), ncol = 1) |> as.vector() |> diag()
G_s <- MinT_G(S, W_s)

# Example base forecasts

```

```
fc <- c(10, 4, 4, 2, 4, 2, 2) # the first base forecast in the middle level performs poorly
```

The two functions (`mip_10` and `.mip_10`) give the identical estimated parameters.

- `mip_10`: use all the objective parameters and constraints, but select different constraints according to the  $\lambda$  values.
- `.mip_10`: use different objective parameters, objective functions, and constraints according to the  $\lambda$  values.

```
print("# mip_10")
rec <- mip_10(fc, S, W_I, GO = NULL,
             lambda_0 = 0, lambda_1 = 0, lambda_2 = 0, M = NULL,
             solver = "gurobi")
rec$G

print("# .mip_10")
.rec <- .mip_10(fc, S, W_I, GO = NULL,
               lambda_0 = 0, lambda_1 = 0, lambda_2 = 0, M = NULL,
               solver = "gurobi")
.rec$G
```

```
## [1] "# mip_10"
##      [,1]      [,2] [,3]      [,4]      [,5] [,6] [,7]
## [1,]    0 0.2380952    0 0.7619048 -0.2380952    0    0
## [2,]    0 0.2380952    0 -0.2380952 0.7619048    0    0
## [3,]    0 -0.0952381    0 0.0952381 0.0952381    1    0
## [4,]    0 -0.0952381    0 0.0952381 0.0952381    0    1
## [1] "# .mip_10"
##      [,1]      [,2] [,3]      [,4]      [,5] [,6] [,7]
## [1,]    0 0.2380952    0 0.7619048 -0.2380952    0    0
## [2,]    0 0.2380952    0 -0.2380952 0.7619048    0    0
## [3,]    0 -0.0952381    0 0.0952381 0.0952381    1    0
## [4,]    0 -0.0952381    0 0.0952381 0.0952381    0    1
```

If we do not put any penalty terms on  $\mathbf{G}$ , that is  $\lambda_0 = \lambda_1 = \lambda_2 = 0$ , the output should be identical to the result given by MinT using the corresponding estimator for the variance-covariance matrix of the base forecast errors.

Here we consider  $W_I$  as an identity matrix and compare various results obtained by our method and MinT.

### 1. $G$ matrix (different)

```
print("# mip_l0")
(G <- rec$G)

print("# MinT (ols)")
(G_ols <- MinT_G(S, W_I))

## [1] "# mip_l0"
##      [,1]      [,2] [,3]      [,4]      [,5] [,6] [,7]
## [1,]    0  0.2380952    0  0.7619048 -0.2380952    0    0
## [2,]    0  0.2380952    0 -0.2380952  0.7619048    0    0
## [3,]    0 -0.0952381    0  0.0952381  0.0952381    1    0
## [4,]    0 -0.0952381    0  0.0952381  0.0952381    0    1
## [1] "# MinT (ols)"
##      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.1428571  0.2380952 -0.0952381  0.61904762 -0.38095238 -0.04761905
## [2,] 0.1428571  0.2380952 -0.0952381 -0.38095238  0.61904762 -0.04761905
## [3,] 0.1428571 -0.0952381  0.2380952 -0.04761905 -0.04761905  0.61904762
## [4,] 0.1428571 -0.0952381  0.2380952 -0.04761905 -0.04761905 -0.38095238
##      [,7]
## [1,] -0.04761905
## [2,] -0.04761905
## [3,] -0.38095238
## [4,]  0.61904762
```

### 2. Reconciled forecasts $\tilde{y}$ (same)

```
# mip_l0, MinT (ols)
cbind(S %*% G %*% fc, S %*% G_ols %*% fc)

##      [,1]      [,2]
## [1,] 9.428571 9.428571
## [2,] 5.047619 5.047619
```

```
## [3,] 4.380952 4.380952
## [4,] 1.523810 1.523810
## [5,] 3.523810 3.523810
## [6,] 2.190476 2.190476
## [7,] 2.190476 2.190476
```

3. Objective values  $(\hat{\mathbf{y}}_h - \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_h)' \mathbf{W}_h^{-1} (\hat{\mathbf{y}}_h - \mathbf{S}\mathbf{G}\hat{\mathbf{y}}_h)$  (same and both satisfy  $\mathbf{G}\mathbf{S} = \mathbf{I}_{n_b}$ )

```
print("# mip_l0")
obj_MP(fc, S, W_I, G)

print("# MinT (ols)")
obj_MP(fc, S, W_I, G_ols)
```

```
## [1] "# mip_l0"
## [1] 2.095238
## [1] "# MinT (ols)"
## [1] 2.095238
```

So, when  $\lambda_0 = \lambda_1 = \lambda_2 = 0$ , our method and MinT give same reconciled forecasts and objective values but **different  $\mathbf{G}$  matrix**. Here are some explanations:

- For our method, the columns of  $\hat{\mathbf{y}}'_h \otimes \mathbf{S}$  are not linearly independent, there are infinitely many least-squares solutions.
- For MinT, under the assumption  $\mathbf{G}\mathbf{S} = \mathbf{I}_{n_b}$ , Wickramasuriya et al. (2019) first derive the explicit solution as  $\mathbf{G} = (\mathbf{S}'\mathbf{W}_h^{-1}\mathbf{S})^{-1} \mathbf{S}'\mathbf{W}_h^{-1}$ , then reformulate the trace minimization problem in terms of a linear equality constrained least squares problem

$$\min_{\tilde{\mathbf{y}}_h} \frac{1}{2} (\hat{\mathbf{y}}_h - \tilde{\mathbf{y}}_h)' \mathbf{W}_h^{-1} (\hat{\mathbf{y}}_h - \tilde{\mathbf{y}}_h), \text{ s.t. } \tilde{\mathbf{y}}_h = \mathbf{S}\tilde{\mathbf{b}}_h.$$

This is a Generalized Least Squares problem, actually the objective parameter is the reconciled forecasts at the bottom level  $\tilde{\mathbf{b}}_h$  rather than  $\mathbf{G}$ .

- The GLS problem is equivalent to the following problem

$$\min_{\tilde{\mathbf{y}}_h} \frac{1}{2} (\hat{\mathbf{y}}_h - \tilde{\mathbf{y}}_h)' \mathbf{W}_h^{-1} (\hat{\mathbf{y}}_h - \tilde{\mathbf{y}}_h), \text{ s.t. } \mathbf{U}'\tilde{\mathbf{y}}_h = \mathbf{0} \text{ where } \mathbf{U}' = [\mathbf{I}_{n-n_b} | -\mathbf{C}].$$

- In order to perform subset selection with shrinkage for  $\mathbf{G}$ . We can reformulate our minimization problem as follows.

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

In this case, if  $\lambda_0 = \lambda_1 = \lambda_2$ , the objective parameters become  $\tilde{\mathbf{b}}$  and we only output the estimation of  $\tilde{\mathbf{b}}$ . But it is actually the same with the original formulation.

### Some questions

1. Check the original objective function (the trace of the variance covariance matrix of the **reconciled forecast errors**) given by  $\text{var}[\mathbf{y}_{t+h} - \tilde{\mathbf{y}}_h | I_t] = \mathbf{S}\mathbf{G}\mathbf{W}\mathbf{G}'\mathbf{S}'$  subject to  $\mathbf{S}\mathbf{G}\mathbf{S} = \mathbf{S}$ . The results are different.

```
print("# mip_l0")
obj_MinT(S, W_I, G)

print("# MinT (ols)")
obj_MinT(S, W_I, G_ols)
```

```
## [1] "# mip_l0"
## [1] 9.428571
## [1] "# MinT (ols)"
## [1] 4
```

If we give some penalty to shrink  $\mathbf{G}$  towards  $\mathbf{G}_{ols}$ , even if the value of  $\lambda_1$  is very small, it will return the same  $\mathbf{G}$  as that obtained from MinT explicit solution.

```
print("# G_ols")
G_ols

print("# mip_l0 with lambda_0 = lambda_1 = lambda_2 = 0")
G
```



```

print("# mip_l0 with lambda_1 = 0.1 and G0 = G_ols")
mip_l0(fc, S, W_I, G0 = G_ols,
       lambda_0 = 0, lambda_1 = 0.1, lambda_2 = 0, M = NULL,
       solver = "gurobi")$G

print("# mip_l0 with lambda_1 = 0.001 and G0 = G_ols")
mip_l0(fc, S, W_I, G0 = G_ols,
       lambda_0 = 0, lambda_1 = 0.001, lambda_2 = 0, M = NULL,
       solver = "gurobi")$G

print("# mip_l0 with lambda_1 = 0.1 and G0 = G_bu")
G_bu <- cbind(rep(0, 4), rep(0, 4), rep(0, 4), diag(1, 4))
mip_l0(fc, S, W_I, G0 = G_bu,
       lambda_0 = 0, lambda_1 = 0.1, lambda_2 = 0, M = NULL,
       solver = "gurobi")$G

print("# mip_l0 with lambda_0 = 0.1, lambda_1 = 0.1 and G0 = G_bu")
mip_l0(fc, S, W_I, G0 = G_bu,
       lambda_0 = 0.1, lambda_1 = 0.1, lambda_2 = 0, M = NULL,
       solver = "gurobi")$G

```

```

## [1] "# G_ols"
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.1428571 0.2380952 -0.0952381 0.61904762 -0.38095238 -0.04761905
## [2,] 0.1428571 0.2380952 -0.0952381 -0.38095238 0.61904762 -0.04761905
## [3,] 0.1428571 -0.0952381 0.2380952 -0.04761905 -0.04761905 0.61904762
## [4,] 0.1428571 -0.0952381 0.2380952 -0.04761905 -0.04761905 -0.38095238
##           [,7]
## [1,] -0.04761905
## [2,] -0.04761905
## [3,] -0.38095238
## [4,] 0.61904762
## [1] "# mip_l0 with lambda_0 = lambda_1 = lambda_2 = 0"
##           [,1]      [,2] [,3]      [,4]      [,5] [,6] [,7]
## [1,] 0 0.2380952 0 0.7619048 -0.2380952 0 0

```

```

## [2,]    0  0.2380952    0 -0.2380952  0.7619048    0    0
## [3,]    0 -0.0952381    0  0.0952381  0.0952381    1    0
## [4,]    0 -0.0952381    0  0.0952381  0.0952381    0    1
## [1] "# mip_l0 with lambda_1 = 0.1 and G0 = G_ols"
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.1428571  0.2380952 -0.0952381  0.61904762 -0.38095238 -0.04761905
## [2,] 0.1428571  0.2380952 -0.0952381 -0.38095238  0.61904762 -0.04761905
## [3,] 0.1428571 -0.0952381  0.2380952 -0.04761905 -0.04761905  0.61904762
## [4,] 0.1428571 -0.0952381  0.2380952 -0.04761905 -0.04761905 -0.38095238
##          [,7]
## [1,] -0.04761905
## [2,] -0.04761905
## [3,] -0.38095238
## [4,]  0.61904762
## [1] "# mip_l0 with lambda_1 = 0.001 and G0 = G_ols"
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.1428571  0.2380952 -0.09523810  0.61904762 -0.38095238 -0.04761905
## [2,] 0.1428571  0.2380952 -0.09523809 -0.38095238  0.61904762 -0.04761905
## [3,] 0.1428571 -0.0952381  0.23809524 -0.04761905 -0.04761905  0.61904762
## [4,] 0.1428571 -0.0952381  0.23809524 -0.04761905 -0.04761905 -0.38095238
##          [,7]
## [1,] -0.04761905
## [2,] -0.04761905
## [3,] -0.38095238
## [4,]  0.61904762
## [1] "# mip_l0 with lambda_1 = 0.1 and G0 = G_bu"
##          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] -0.13285649  0.2130952  0.13285649  0.91976125 -0.08023875  5.384103e-10
## [2,] -0.13313737  0.2130952  0.13313737 -0.07995787  0.92004213  5.368003e-10
## [3,]  0.04060371 -0.0702381 -0.04060371  0.02963438  0.02963438  1.000000e+00
## [4,]  0.04060039 -0.0702381 -0.04060039  0.02963770  0.02963770 -2.210553e-09
##          [,7]
## [1,]  5.384103e-10
## [2,]  5.368003e-10
## [3,] -2.209846e-09
## [4,]  1.000000e+00

```

```

## [1] "# mip_10 with lambda_0 = 0.1, lambda_1 = 0.1 and G0 = G_bu"
##      [,1]      [,2] [,3]      [,4]      [,5] [,6] [,7]
## [1,]    0  0.2130952    0  0.7869048 -0.2130952    0    0
## [2,]    0  0.2130952    0 -0.2130952  0.7869048    0    0
## [3,]    0 -0.0702381    0  0.0702381  0.0702381    1    0
## [4,]    0 -0.0702381    0  0.0702381  0.0702381    0    1

```

2. A.2 Theorem1 in Wickramasuriya et al. (2019).

**Lemma 1 of Wang et al.(1986):**

Let  $K$  and  $S$  be  $(n \times n)$  matrices. If  $K = K' \geq 0$  and  $S$  is symmetric, then

$$\lambda_{\min}(S) \operatorname{tr}(K) \leq \operatorname{tr}(KS) \leq \lambda_{\max}(S) \operatorname{tr}(K)$$

The objective function,

$$\operatorname{tr}[SPW_h P' S'] = \operatorname{tr}[S' SPW_h P'] .$$

$S'S$  and  $PW_h P'$  are both symmetric, and positive definite and positive semi-definite respectively. So we have

$$\lambda_{\min}(S'S) \operatorname{tr}[PW_h P'] \leq \operatorname{tr}[S' SPW_h P'] \leq \lambda_{\max}(S'S) \operatorname{tr}[PW_h P'] ,$$

where  $S'S$  is given and its eigenvalues are positive. So the original minimization problem can be restated as

$$\min_P \lambda_{\max}(S'S) \operatorname{tr}[PW_h P'] \quad \text{s.t. } PS = I \implies \min_P \operatorname{tr}[PW_h P'] \quad \text{s.t. } PS = I$$

- The choice of  $\lambda$  is scale dependent.
- One-step head forecast VS. multi-step ahead forecast.
- Different models to be used for each series.
- Computationally expensive.

#### 4.4.2 Hyperparameter tuning

Minimization problem:

$$\begin{aligned}
& \min_G \frac{1}{2} (\hat{\mathbf{y}}_h - (\hat{\mathbf{y}}'_h \otimes \mathbf{S}) \operatorname{vec}(\mathbf{G}))' \mathbf{W}_h^{-1} (\hat{\mathbf{y}}_h - (\hat{\mathbf{y}}'_h \otimes \mathbf{S}) \operatorname{vec}(\mathbf{G})) \\
& \quad + \lambda_0 \sum_{j=1}^n \|\mathbf{G}_{:,j}\|_0 + \lambda_1 \left\| \operatorname{vec}(\mathbf{G} - \mathbf{G}^0) \right\|_1 + \lambda_2 \left\| \operatorname{vec}(\mathbf{G} - \mathbf{G}^0) \right\|_2^2 \\
& \text{s.t. } \mathbf{GS} = \mathbf{I}_{n_b}
\end{aligned}$$

- Set  $\lambda_1$  and  $\lambda_2$  to zero.
- Choose  $\lambda_{\max}$  to achieve maximum subset selection.
  - unbiasedness assumption  $\text{rank}(\mathbf{G}) = n_b \implies \min \sum_{j=1}^n \|\mathbf{G}_{\cdot j}\|_0 = n_b$
  - assume that the maximum loss of the first term is  $\frac{1}{2} \hat{\mathbf{y}}_h' \mathbf{W}_h^{-1} \hat{\mathbf{y}}_h$
- Set  $\lambda_{\max}$  to  $\frac{1}{n_b} \frac{1}{2} \hat{\mathbf{y}}_h' \mathbf{W}_h^{-1} \hat{\mathbf{y}}_h$ .
- Similarly to the `glmnet` package, we select a minimum value  $\lambda_{\min} = \epsilon \lambda_{\max}$ , and construct a sequence of  $K - 1$  values of  $\lambda$  decreasing from  $\lambda_{\max}$  to  $\lambda_{\min}$  **on the log scale**.  $\epsilon = 10^{-4}$  and  $K = 20$ .
- Select the best value of  $\lambda_0$  from  $0, \lambda_{\min}, \dots, \lambda_{\max}$  by minimizing the sum of squared reconciled forecast errors in the training set, even though fitted values are often not true one-step ahead forecasts. (Avoid cross-validation)

**Simulation setup:** same with sec 3.4 of Wickramasuriya et al. (2019). This process is repeated 500 times.

#### Simulation results:

- Scenario 1: models for some bottom-level series (AA and BB) are mis-specified using drift method

Total	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
BU	47.347	37.313	38.627	5.086	4.011	4.156
MinTc	9.559	10.451	12.717	1.012	1.109	1.354
MinTc_subse	9.665	10.499	12.752	1.022	1.115	1.358
MinTs	9.535	10.434	12.701	1.009	1.107	1.352
MinTs_subse	9.526	10.444	12.730	1.009	1.107	1.354
OLS	11.869	11.862	13.898	1.261	1.261	1.482
OLS_subset	11.016	11.197	13.314	1.168	1.191	1.421
WLSs	18.583	16.400	18.080	1.988	1.753	1.936
WLSs_subse <sup>1</sup>	19.084	15.892	17.597	2.045	1.703	1.890
WLSv	9.747	10.514	12.743	1.030	1.115	1.356
WLSv_subse	9.534	10.460	12.706	1.009	1.109	1.352
base	9.606	10.391	12.649	1.016	1.102	1.346

Middle	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
BU	33.430	26.191	27.162	5.320	4.173	4.332
MinTc	6.446	7.186	8.850	1.012	1.136	1.404
MinTc_subse	6.450	7.198	8.865	1.013	1.139	1.407
MinTs	6.427	7.172	8.836	1.009	1.134	1.402
MinTs_subse	6.477	7.230	8.903	1.017	1.142	1.411
OLS	10.808	9.891	11.244	1.711	1.570	1.789
OLS_subset	10.284	9.580	10.990	1.624	1.516	1.745
WLSs	15.614	13.216	14.384	2.476	2.101	2.291
WLSs_subse1	16.174	13.178	14.386	2.567	2.098	2.294
WLSv	6.650	7.310	8.939	1.042	1.156	1.418
WLSv_subse	6.394	7.214	8.859	1.005	1.141	1.405
base	6.335	7.134	8.811	0.996	1.129	1.398

Bottom	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
BU	18.611	15.149	16.118	4.430	3.614	3.847
MinTc	4.310	4.956	6.156	1.020	1.176	1.463
MinTc_subse	4.290	4.995	6.192	1.015	1.186	1.472
MinTs	4.300	4.947	6.148	1.017	1.174	1.461
MinTs_subse	4.321	5.005	6.212	1.021	1.186	1.475
OLS	17.026	13.765	14.531	4.050	3.281	3.465
OLS_subset	14.290	11.233	12.136	3.397	2.669	2.886
WLSs	17.084	13.832	14.624	4.065	3.298	3.489
WLSs_subse1	16.670	12.808	13.655	3.969	3.054	3.258
WLSv	4.807	5.273	6.412	1.136	1.250	1.524
WLSv_subse	4.326	5.082	6.272	1.024	1.204	1.489
base	18.611	15.149	16.118	4.430	3.614	3.847

All	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
BU	26.950	21.470	22.489	4.778	3.830	4.030
MinTc	5.670	6.378	7.863	1.017	1.155	1.431
MinTc_subse	5.675	6.411	7.893	1.016	1.162	1.437
MinTs	5.655	6.366	7.852	1.014	1.153	1.429
MinTs_subse	5.681	6.418	7.912	1.018	1.163	1.440
OLS	14.513	12.386	13.502	2.983	2.504	2.703
OLS_subset	12.678	10.756	11.977	2.572	2.129	2.350
WLSs	16.878	14.023	15.049	3.314	2.735	2.925
WLSs_subse1	16.873	13.354	14.427	3.294	2.588	2.787
WLSv	6.039	6.604	8.039	1.094	1.204	1.470
WLSv_subse	5.661	6.459	7.930	1.016	1.172	1.445

All	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
base	13.817	12.179	13.535	2.961	2.545	2.790

- Scenario 2: models for top-level series (Total) are mis-specified using drift method

Total	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
BU	9.511	10.466	12.744	1.005	1.110	1.356
MinTc	9.507	10.458	12.723	1.003	1.108	1.353
MinTc_subse	9.471	10.468	12.738	1.001	1.110	1.355
MinTs	9.504	10.450	12.723	1.004	1.108	1.353
MinTs_subse	9.510	10.462	12.735	1.005	1.109	1.355
OLS	38.621	31.122	32.445	4.150	3.332	3.474
OLS_subset	25.452	19.112	21.035	2.757	2.052	2.261
WLSs	24.018	20.512	22.014	2.576	2.192	2.355
WLSs_subse1	9.531	10.426	12.684	1.009	1.105	1.350
WLSv	9.592	10.504	12.764	1.015	1.113	1.358
WLSv_subse	9.550	10.450	12.713	1.010	1.108	1.352
base	65.813	51.137	52.542	7.078	5.478	5.629

Middle	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
BU	6.317	7.149	8.820	0.993	1.131	1.400
MinTc	6.375	7.167	8.846	1.001	1.133	1.403
MinTc_subse	6.363	7.164	8.844	1.000	1.133	1.403
MinTs	6.378	7.159	8.837	1.002	1.132	1.402
MinTs_subse	6.379	7.162	8.844	1.003	1.133	1.403
OLS	19.736	16.458	17.512	3.150	2.618	2.788
OLS_subset	13.879	11.155	12.604	2.227	1.779	2.012
WLSs	12.682	11.415	12.640	2.022	1.814	2.011
WLSs_subse1	6.322	7.135	8.800	0.994	1.129	1.396
WLSv	6.348	7.156	8.821	0.998	1.132	1.400
WLSv_subse	6.356	7.154	8.816	0.999	1.132	1.399
base	6.335	7.134	8.811	0.996	1.129	1.398

Bottom	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
BU	4.199	4.903	6.107	0.995	1.164	1.452
MinTc	4.240	4.920	6.125	1.005	1.168	1.456
MinTc_subse	4.236	4.921	6.129	1.004	1.168	1.457

Bottom	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
MinTs	4.222	4.912	6.120	1.000	1.166	1.455
MinTs_subse	4.229	4.920	6.130	1.002	1.168	1.458
OLS	10.248	9.002	9.846	2.443	2.142	2.344
OLS_subset	7.716	6.715	7.792	1.847	1.600	1.857
WLSs	6.947	6.680	7.669	1.653	1.588	1.825
WLSs_subse <sup>1</sup>	4.218	4.906	6.110	0.999	1.164	1.452
WLSv	4.217	4.909	6.111	0.999	1.165	1.453
WLSv_subse	4.219	4.915	6.120	1.000	1.167	1.455
base	4.199	4.903	6.107	0.995	1.164	1.452

All	RMSE			MASE		
Method	h=1	1-8	1-16	h=1	1-8	1-16
BU	5.563	6.340	7.830	0.996	1.147	1.423
MinTc	5.603	6.353	7.845	1.004	1.150	1.426
MinTc_subse	5.592	6.354	7.849	1.002	1.150	1.427
MinTs	5.593	6.345	7.840	1.001	1.148	1.425
MinTs_subse	5.598	6.353	7.849	1.003	1.150	1.427
OLS	17.012	14.292	15.265	2.889	2.448	2.632
OLS_subset	12.011	9.754	11.058	2.085	1.716	1.959
WLSs	11.024	10.009	11.139	1.891	1.739	1.954
WLSs_subse <sup>1</sup>	5.578	6.332	7.818	0.999	1.146	1.422
WLSv	5.594	6.350	7.836	1.001	1.148	1.424
WLSv_subse	5.591	6.346	7.832	1.001	1.148	1.424
base	13.611	12.145	13.513	1.864	1.770	2.033

1. Model misspecification (also exploring the effect of correlation)
2. Poor forecastability
3. Large hierarchy

#### 4.4.3 Adjustment of the MinT estimates

##### 4.4.3.1 MinT problem

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

$\Downarrow$

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

#### 4.4.3.2 Subset selection problem

$$\min_{\mathbf{G}, \mathbf{A}} \frac{1}{2} (\hat{\mathbf{y}}_h - \mathbf{S} \mathbf{G} \mathbf{A} \hat{\mathbf{y}}_h)' \mathbf{W}_h^{-1} (\hat{\mathbf{y}}_h - \mathbf{S} \mathbf{G} \mathbf{A} \hat{\mathbf{y}}_h) + \lambda_0 \|\mathbf{A} \mathbf{1}\|_0 \quad \text{s.t. } \mathbf{G} \mathbf{A} \mathbf{S} = \mathbf{I}_{n_b}$$

$\Downarrow$  assuming  $\mathbf{G}$  is given by MinT with the  $\mathbf{W}$  estimate

$$\min_{\mathbf{A}} \frac{1}{2} (\hat{\mathbf{y}}_h - (\hat{\mathbf{y}}_h' \otimes (\mathbf{S} \mathbf{G})) \text{vec}(\mathbf{A}))' \mathbf{W}_h^{-1} (\hat{\mathbf{y}}_h - (\hat{\mathbf{y}}_h' \otimes (\mathbf{S} \mathbf{G})) \text{vec}(\mathbf{A})) + \lambda_0 \|\mathbf{A} \mathbf{1}\|_0 \quad \text{s.t. } \mathbf{G} \mathbf{A} \mathbf{S} = \mathbf{I}_{n_b}$$

where  $\mathbf{A}$  is an  $n \times n$  **diagonal matrix**.

The mixed integer optimization (MIO) formulation is

$$\begin{aligned} \min_{\mathbf{A}, \mathbf{z}} \quad & \frac{1}{2} (\hat{\mathbf{y}}_h - (\hat{\mathbf{y}}_h' \otimes (\mathbf{S} \mathbf{G})) \text{vec}(\mathbf{A}))' \mathbf{W}_h^{-1} (\hat{\mathbf{y}}_h - (\hat{\mathbf{y}}_h' \otimes (\mathbf{S} \mathbf{G})) \text{vec}(\mathbf{A})) + \lambda_0 \sum_{j=1}^n z_j \\ \text{s.t.} \quad & \mathbf{G} \mathbf{A} \mathbf{S} = \mathbf{I}_{n_b} \Leftrightarrow (\mathbf{S}' \otimes \mathbf{G}) \text{vec}(\mathbf{A}) = \text{vec}(\mathbf{I}_{n_b}) \\ & a_{ij} = 0, \quad i \neq j \\ & -M\mathbf{z} \leq \mathbf{A} \mathbf{1} \leq M\mathbf{z} \\ & z_j \in \{0, 1\} \\ & i, j \in [n] \end{aligned}$$

Compared with the method that directly estimates all the elements of  $\mathbf{G}$ , this method leads to the adjustment of the estimated MinT  $\mathbf{G}$ . Specifically, the added penalty will zero out some columns of  $\mathbf{G}_{MinT}$  and scale remaining columns proportionally. Elements in the same column will be scaled by the same proportion, which means the same scale is applied to the same base forecast to obtain the reconciled bottom forecasts through forecast combination.

$$\mathbf{G} = [\mathbf{g}_{\cdot 1}, \mathbf{g}_{\cdot 2}, \dots, \mathbf{g}_{\cdot n}]$$

$$\mathbf{G} \mathbf{A} = [a_1 \mathbf{g}_{\cdot 1}, a_2 \mathbf{g}_{\cdot 2}, \dots, a_n \mathbf{g}_{\cdot n}]$$

- It will always return an identity matrix  $\mathbf{A}$  due to unbiasedness assumption.

## 5 Other issues to be solved

### 5.1 Temporal reconciliation

- Remove all nodes in a level or several levels.

### 5.2 Non-negative constraints