Time-Varying Graphical Lasso: Summary and Application

Alaa Bouattour Rayen Ben Hassen
Mahdi Ben Ayed Ola Hamad

Contents

1 Article Summarization and Theoretical Formulation

1.1 Introduction

In many applications, such as finance, neuroscience, and social networks, relationships between variables evolve over time. Modeling these dynamics is critical for understanding and predicting system behavior. The **Time-Varying Graphical Lasso (TVGL)** extends the classical Graphical Lasso to infer sparse precision matrices (inverse covariance matrices) that vary smoothly over time. This allows for the simultaneous discovery of network structures and their temporal evolution.

Motivation and Setting:

- Standard Graphical Lasso assumes data are independent and identically distributed (i.i.d.) and estimates a single sparse precision matrix Θ.
- In time-dependent settings, relationships among variables evolve, requiring a framework that captures temporal variations while preserving sparsity.
- TVGL addresses this by introducing temporal penalties to encourage smooth transitions in precision matrices.

Applications:

- *Finance*: Evolving correlations among stocks under changing market conditions.
- *Neuroscience*: Brain connectivity changes with cognitive tasks or disease progression.
- *Social Networks*: Shifting user interactions over time.

1.2 Theoretical Importance of the Inverse Covariance Matrix

The inverse covariance matrix, also known as the precision matrix, plays a crucial role in understanding the relationships among variables in multivariate data. It encodes the conditional independence structure of the variables:

Conditional Independence: If the (i, j)-th entry of the precision matrix $\Theta = \Sigma^{-1}$ is zero, i.e., $\Theta_{ij} = 0$, then the variables X_i and X_j are conditionally independent given all other variables in the system. Mathematically,

$$\Theta_{ij} = 0 \quad \iff \quad P(X_i, X_j | X_{\setminus\{i,j\}}) = P(X_i | X_{\setminus\{i,j\}}) P(X_j | X_{\setminus\{i,j\}}). \tag{1}$$

This makes the precision matrix a critical tool for understanding and visualizing dependencies in a network.

Proof: Given a multivariate Gaussian distribution $X \sim \mathcal{N}(\mu, \Sigma)$, the joint density is:

$$f_X(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right). \tag{2}$$

The conditional density of X_i given $X_{\setminus i}$ can be derived from the joint Gaussian density. The conditional independence between X_i and X_j given $X_{\setminus\{i,j\}}$ follows directly from the structure of Σ^{-1}:

- If $\Theta_{ij} = 0$, the (i, j) entry in the inverse covariance matrix, this implies that there is no direct dependency between X_i and X_j when conditioned on the remaining variables.
- This property simplifies network interpretation by representing the dependencies through the graph encoded by Θ.

Real-World Implications: In practice, this property of the precision matrix is used in:

- *Neuroscience:* Identifying functional connectivity in the brain by analyzing relationships between different regions over time.
- *Finance:* Understanding dependencies between assets and mitigating risk by identifying conditional independencies.
- *Genomics:* Mapping gene interactions by inferring sparse dependencies between genes.

1.3 Theoretical Formulation

1.3.1 Problem Definition and Objective

Given a sequence of empirical covariance matrices $\{S_t\}_{t=1}^T$ computed from time-dependent data, the goal is to estimate precision matrices $\{\Theta_t\}_{t=1}^T$ for each time step t. The objective combines likelihood-based data fidelity, sparsity, and temporal smoothness:

$$\min_{\Theta_1,\ldots,\Theta_T \succ 0} \quad \sum_{t=1}^{T} \Big[-n_t \log\det(\Theta_t) + n_t \mathrm{Tr}(S_t\Theta_t) \Big] + \lambda \sum_{t=1}^{T} \|\Theta_t\|_{\text{offdiag},1} + \beta \sum_{t=2}^{T} \psi(\Theta_t - \Theta_{t-1}), \tag{3}$$

where:

- $-n_t \log\det(\Theta_t) + n_t \mathrm{Tr}(S_t\Theta_t)$: Negative log-likelihood for Gaussian data.
- $\|\Theta_t\|_{\text{offdiag},1}$: Encourages sparsity by penalizing off-diagonal elements.
- $\psi(\cdot)$: Temporal penalty encouraging smooth transitions between consecutive matrices.
- λ, β: Regularization parameters controlling sparsity and smoothness.

1.3.2 Choices for Temporal Penalty ψ

Different penalties model various evolutionary patterns:

- **Element-wise L1**: $\psi(X) = \|X\|_1$ for sparse edge changes.
- **Group Lasso (L2)**: $\psi(X) = \sum_j \|X_{:,j}\|_2$, enforcing node-wise changes.
- **Laplacian**: $\psi(X) = \|X\|_F^2$, favoring smooth variations.
- **Infinity Norm**: $\psi(X) = \|X\|_\infty$ for block-wise changes.
- **Perturbed Node**: A row-column overlap norm allowing single-node rewiring.

Advantages of Temporal Penalties:

- **Sparsity**: Ensures the precision matrices are interpretable by promoting zero elements.
- **Smooth Transitions**: Captures gradual changes in relationships, reflecting real-world dynamics.
- **Flexibility**: Different penalties allow modeling various types of network evolution.

1.4 Optimization Approach

The TVGL optimization problem can be solved using:

- **Interior-point methods**: Effective for small-scale problems.
- **First-order methods (e.g., ADMM)**: Scalable for large p or T by solving subproblems iteratively.

ADMM-based Approach:

- Splits the problem into smaller subproblems for each Θ_t and temporal penalty term.
- Ensures convergence to the global optimum with computational efficiency.
- Adapts well to high-dimensional settings by leveraging sparsity.

Challenges:

- High computational cost for large T or p.
- Balancing sparsity and smoothness parameters (λ, β) requires careful tuning.

2 Implementation Details and Testing on Synthetic Data

2.1 Implementation Details

In this section, we describe the approach taken to solve the time-varying graphical lasso (TVGL) problem. Since our problem dimension is of moderate size (both in terms of the number of variables p and the number of time steps T), we formulate the entire problem—including the log-determinant likelihood term, the ℓ_1 sparsity penalties, and the temporal smoothness penalties—as a single convex program. We then use CVXPY to express all constraints (particularly the positive-semidefinite constraint on Θ_i) and rely on the `SCS` solver to handle the exponential cone representation of $\log\det(\Theta_i)$.

The solver `SCS` is a first-order method. Although it can be slower or less precise than specialized interior-point methods, it works out of the box for exponential cone constraints, enabling us to solve $\log\det(\Theta)$ problems.

Features and Extensions:

- *Multiple Smoothness Penalties* – We can specify different $\psi(\Theta_i - \Theta_{i-1})$ functions, such as element-wise L_1 penalties or the "perturbed node" penalty, simply by returning the corresponding CVXPY expression (plus any auxiliary constraints).
- *Hyperparameter Tuning* – The parameters λ (sparsity) and β (smoothness) can be chosen via cross-validation or other model selection criteria such Akaike Information Criterion (AIC) to balance within-timestep fit and across-timestep regularity.
- *Scalability* – For larger problems, a block-coordinate or ADMM approach might outperform a single-shot conic solve, but that is beyond our current scope.

2.2 Synthetic Data Generation: Market Dynamics and Conditional Independence

2.2.1 Objective

The synthetic data aims to simulate market-like dynamics with two groups of stocks and their associated indices. The primary goal is to analyze conditional independence relationships, introduce a market event that shifts these dynamics, and test the ability of TVGL to estimate the evolving precision matrices.

2.2.2 Setup and Dynamics

Initial Configuration

- **Groups and Variables:**
  - **Group 1:** 3 stocks and 1 index associated with these stocks.
  - **Group 2:** 2 stocks and 1 index associated with these stocks.
- **Dynamic Relationships:**
  - **Indexes:** Correlated and conditionally correlated. Knowing one index provides additional information about the other.
  - **Stocks within a group:** Correlated and conditionally correlated. Knowing one stock provides information about others in the same group.
  - **Stocks between groups:** Conditionally independent. Stocks in different groups are independent given the indices and other stocks in their respective groups.
- **Matrix Constraints:**
  - Precision matrix (inverse covariance matrix) reflects the dynamics, including zeros for conditional independence.
  - Must be positive definite for invertibility.

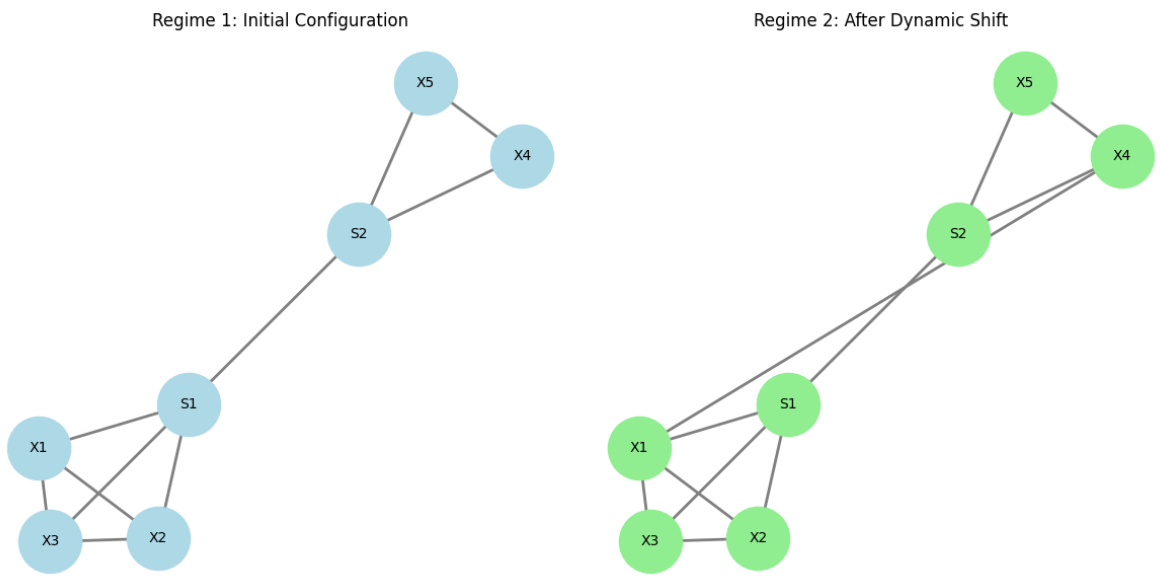

Figure 1: Your caption here

Market Change: Partial Acquisition Scenario

- **Dynamic Shift:** Partial acquisition between Group 1 and Group 2 changes the dynamics.
  - Stocks in the two groups are no longer conditionally independent.
  - Knowing a stock in one group provides additional information about stocks in the other group.
- **Matrix Adjustment:**
  - Update the precision matrix: Add non-zero coefficients for newly dependent variables.
  - Ensure no changes to other zero coefficients to preserve the original conditional independence relationships.
  - Maintain positive definiteness to ensure invertibility.

2.3 Implementation Plan

Step 1: Generate Initial Precision Matrix Construct the precision matrix to reflect the initial relationships:

- Correlations between stocks and indexes within the same group.
- Conditional independence between groups.

Ensure positive definiteness of the precision matrix.

Step 2: Compute Covariance Matrix Compute the covariance matrix analytically as the inverse of the precision matrix.

Step 3: Generate Synthetic Data Use the covariance matrix to generate multivariate normal data for time horizon T.

Step 4: Modify Precision Matrix Adjust the precision matrix to reflect the new market dynamics:

- Add non-zero coefficients for newly dependent variables.
- Preserve zero coefficients that correspond to conditional independence.
- Ensure the matrix remains positive definite and invertible.

Step 5: Generate Data for Updated Dynamics Compute the updated covariance matrix. Generate synthetic data for horizon T'.

Step 6: Concatenate Data and Estimate Relationships Concatenate data from the two horizons. Estimate the covariance matrices for both regimes using the synthetic data. Compute the precision matrices from the estimated covariance matrices.

Step 7: Analyze Results Identify pairs of variables that are conditionally independent in each regime. Compare the results to the theoretical precision matrices and highlight discrepancies where estimation produces incorrect results.

2.4 Testing and Results

To test the correctness and robustness of the algorithm, we created two datasets based on the principles previously discussed. Both datasets are derived from two true inverse covariance matrices.

Matrix 1:

$$K_1 = \begin{bmatrix} 2.0 & 0.2 & 0.2 & 0.3 & 0.0 & 0.0 & 0.0 \\ 0.2 & 2.0 & 0.2 & 0.3 & 0.0 & 0.0 & 0.0 \\ 0.2 & 0.2 & 2.0 & 0.3 & 0.0 & 0.0 & 0.0 \\ 0.3 & 0.3 & 0.3 & 2.0 & 0.0 & 0.0 & 0.2 \\ 0.0 & 0.0 & 0.0 & 0.0 & 2.0 & 0.2 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.2 & 2.0 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.2 & 0.3 & 0.3 & 2.0 \end{bmatrix}$$

Matrix 2:

$$K_2 = \begin{bmatrix} 2.0 & 0.2 & 0.2 & 0.3 & 0.3 & 0.0 & 0.0 \\ 0.2 & 2.0 & 0.2 & 0.3 & 0.0 & 0.0 & 0.0 \\ 0.2 & 0.2 & 2.0 & 0.3 & 0.0 & 0.0 & 0.0 \\ 0.3 & 0.3 & 0.3 & 2.0 & 0.0 & 0.0 & 0.2 \\ 0.3 & 0.0 & 0.0 & 0.0 & 2.0 & 0.2 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.2 & 2.0 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.2 & 0.3 & 0.3 & 2.0 \end{bmatrix}$$

2.4.1 First Dataset: True Permanent Change

The first dataset represents a true permanent change in the market dynamics between two specific nodes. We generated 80,000 historical points from K_1 followed by 120,000 points from K_2. To align with the methodology, we consider each 20,000 points as one period. Therefore, the vector of true periods is:

$$[K_1, K_1, K_1, K_1, K_2, K_2, K_2, K_2, K_2, K_2]$$

We expect a change in the estimated covariance matrix between the 4th and 5th periods. The testing code details can be found in the `synthetic_data_generatinon_and_testing.ipynb` notebook.

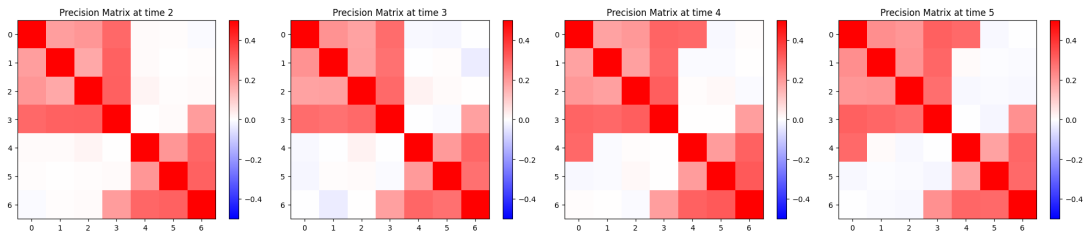

Figure 2: Estimated Matrices for True Permanent Change

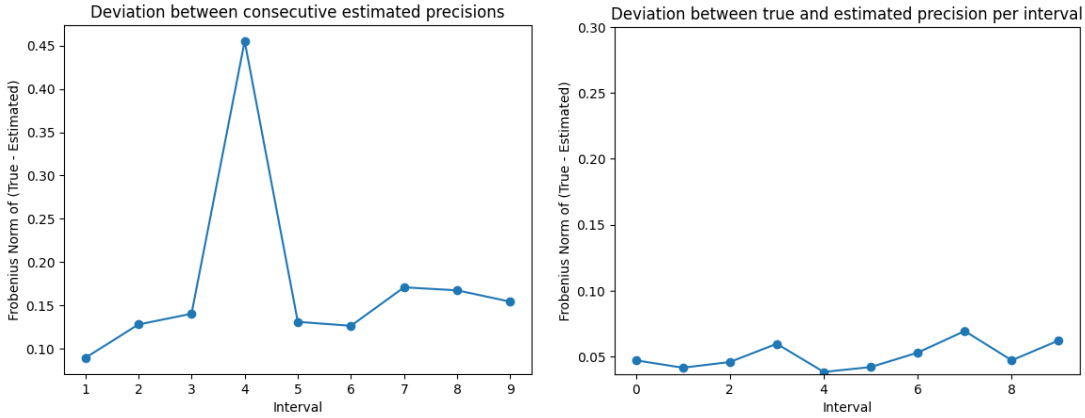

Figure 3: Difference Between True and Estimated Matrices

We observe that the estimated matrices are very close to the true matrices, with a difference in norm around 8% for all matrices. Additionally, we notice a peak in the curve of $\|\Theta(i) - \Theta(i-1)\|$ at the 5th period, indicating a change in market dynamics. This aligns with real-world scenarios where such changes signify significant market shifts.

2.4.2 Second Dataset: Temporary Perturbation

The second dataset represents a temporary perturbation in market dynamics. Here, one period (20,000 points) is governed by K_2, while all other periods follow K_1. The vector of true periods is:

$$[K_1, K_1, K_1, K_1, K_2, K_1, K_1, K_1, K_1, K_1]$$

In this case, we expect the system to be robust to this noise and maintain a consistent matrix structure.

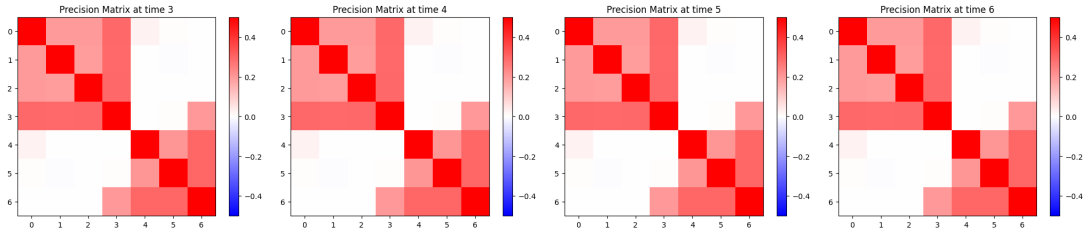

Figure 4: Estimated Matrices for Temporary Perturbation

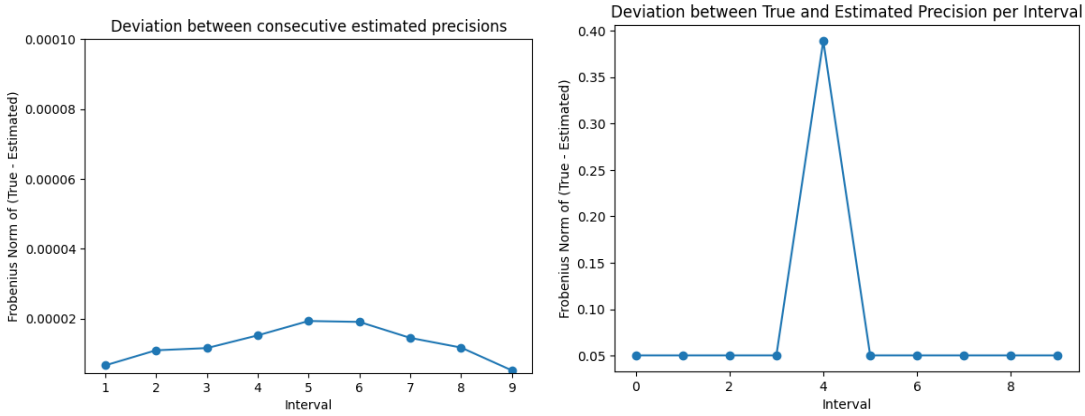

Figure 5: Difference Between True and Estimated Matrices for Temporary Perturbation

We observe that the estimated matrices remain consistent apart from the temporary perturbation in the 5th period. Moreover, the curve of $\|\Theta(i) - \Theta(i-1)\|$ remains stable, effectively avoiding a false detection of market changes. This demonstrates the system's robustness to noise in real-world scenarios.

3 Event detection: S&P500 case

3.1 Overview

By examining historical stock prices, we can infer a financial network to model relationships between different companies. Learning the structure of this network is useful because it allows us to model how certain stocks are related. This can be leveraged to predict future prices, understand economic trends, or diversify a portfolio by avoiding highly correlated stocks. In this study, we infer this network by treating the daily closing price of each stock as a sensor observation and applying the TVGL framework to identify structural changes.

3.2 Single-Stock Perturbations

To detect localized events, we observed daily stock prices for six large American companies: Apple, Google, Amazon, Intel, Boeing, and FedEx. Generally, large companies exhibit a stable correlation network; however, sudden events can cause shifts in this structure. The TVGL model allows for the detection of such events by incorporating a temporal dynamic that penalizes changes affecting single nodes. This approach reflects situations where a specific company experiences a significant perturbation, leaving the rest of the network relatively unchanged. By solving the TVGL optimization problem with this node-perturbation penalty, we identified one major event during the dataset's time period, which corresponded to a significant shift in Apple's connections. Further investigation revealed that this event coincided with Apple's announcement of the original iPad.

3.3 Large-Scale Events

Beyond single-stock perturbations, the TVGL framework can be extended to detect macro-level events affecting the entire market. By applying the model to the S&P 500 dataset, we analyzed structural shifts in the network of the 6 major companies. The maximum temporal deviation, identified through an ℓ_1 penalty, corresponded to the period April-Mai 2010. This period aligns with the infamous "Flash Crash," during which the market experienced a sudden and dramatic drop of 9% before quickly rebounding.

3.4 Results

The deviations between consecutive precision matrices were plotted to detect significant events. A peak in the deviation plot indicates a potential event. As shown in Figure 6, a significant deviation occurred around mid-December 2009/ January 2010, suggesting a structural change in the covariance structure of stock returns (the single-stock perturbation example). Another significant deviation occured around April/Mai 2010, which consistent with the "Flash Crash" (Large-Scale events example).

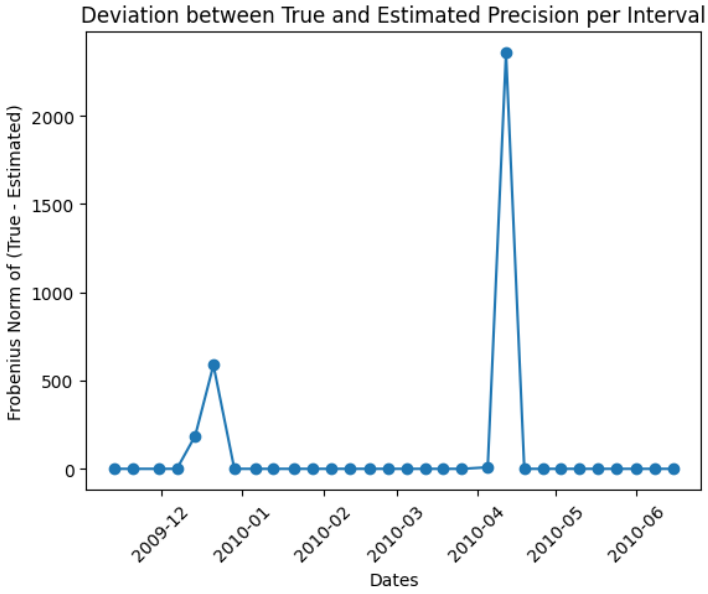

Figure 6: Deviation between True and Estimated Precision per Interval

4 Trading Strategy Comparison: Inverse Covariance vs. Time-Varying Graphical Lasso (TVGL)

4.1 Overview

In portfolio optimization, accurately estimating the precision matrix, which is the inverse of the covariance matrix, is crucial for determining optimal asset weights. Two prevalent methods for precision matrix estimation are the traditional Inverse Covariance approach and the Time-Varying Graphical Lasso (TVGL). This section presents a comparative analysis of these two methods within a trading strategy framework, highlighting their performance based on cumulative returns, Sharpe Ratios, and Maximum Drawdowns.

4.2 Methodology

4.2.1 Data Acquisition and Preparation

The dataset comprises daily adjusted closing prices for six major stocks: Apple Inc. (`AAPL`), Alphabet Inc. (`GOOG`), Amazon.com Inc. (`AMZN`), Intel Corporation (`INTC`), The Boeing Company (`BA`), and FedEx Corporation (`FDX`). The data spans from August 1, 2010, to August 1, 2015.

- **Data Source**: Yahoo Finance via the `yfinance` library.
- **Return Calculation**: Daily returns are computed as percentage changes of the adjusted closing prices.

4.2.2 Precision Matrix Estimation Methods

1. Inverse Covariance The Inverse Covariance method involves calculating the empirical covariance matrix of asset returns over a rolling window and then inverting it to obtain the precision matrix. A small regularization term ($\lambda = 10^{-4}$) is added to the diagonal of the covariance matrix to ensure numerical stability and invertibility.

$$\mathbf{\Theta} = (S + \lambda \mathbf{I})^{-1}$$

To calculate this matrix, we used a window of 25 trading days. So, S in the above equation is the covariance matrix for the period of 25 days, and Θ is our approximation of the inverse of the covariance matrix.

2. Time-Varying Graphical Lasso (TVGL) TVGL extends the graphical lasso by allowing the precision matrix to evolve over time, capturing temporal dependencies and structural changes in the asset relationships. The method estimates a sequence of precision matrices ($\mathbf{\Theta}_t$) over overlapping data segments, enforcing sparsity and smoothness through regularization parameters λ and β.

$$\mathbf{\Theta}_t = \arg\min_{\mathbf{\Theta}} \left\{ -\log\det(\mathbf{\Theta}) + \text{trace}(\mathbf{\Sigma}_t\mathbf{\Theta}) + \lambda\|\mathbf{\Theta}\|_1 + \beta\|\mathbf{\Theta}_t - \mathbf{\Theta}_{t-1}\|_F^2 \right\}$$

The optimization problem described above is calculated over a period of 125 trading days. The duration between two consecutive Θ_i is 25 days, consistent with the period used for the inverse covariance matrix method. The goal is to compare two comparable matrices.

Our precision matrix, Θ, is the final Θ_i generated in this process. The key idea is that this precision matrix should approximate the inverse covariance matrix. However, it is "regularized" and incorporates temporal evolution information of Θ over the broader period of 125 days.

4.2.3 Portfolio Weight Calculation

For each precision matrix estimated, the portfolio weights are derived using the minimum variance approach:

$$\mathbf{w} = \frac{\mathbf{\Theta 1}}{\mathbf{1}^\top \mathbf{\Theta 1}}$$

where $\mathbf{1}$ is a vector of ones. This ensures that the weights sum to unity, adhering to the full investment constraint.

4.2.4 Trading Strategy Execution

The trading strategy operates as follows:

1. **Rolling Window**: Starting from the `test_date` (day 251), the strategy uses a rolling window of past returns (`window_size`) to estimate the precision matrix ((`window_size`) is 25 days for the Inverse Covariance method and is 125 days for TVGL method).
2. **Weight Allocation**: Based on the estimated precision matrix, portfolio weights are calculated.
3. **Return Calculation**: These weights are applied to the subsequent `step_size`=25 days to compute portfolio returns by compounding daily returns.
4. **Iteration**: The window slides forward by `step_size` days, and the process repeats until the end of the dataset.

4.3 Results

The performance of the two precision matrix estimation methods is summarized in Table 1.

Table 1: Performance Metrics Comparison

Method	Cumulative Return	Sharpe Ratio	Maximum Drawdown
Inverse Covariance	1.429987	1.703372	0.080481
TVGL	1.737669	1.878408	0.098915

4.3.1 Interpretation of Metrics

1. Cumulative Return

- **Inverse Covariance**: A cumulative return of 1.429987 indicates a 142.9987% return over the backtesting period.
- **TVGL**: A cumulative return of 1.737669 signifies a 173.7669% return, outperforming the Inverse Covariance method.

2. Sharpe Ratio The Sharpe Ratio measures risk-adjusted returns. Values above 1 are considered good, with higher values indicating better performance relative to risk.

- **Inverse Covariance**: 1.703372
- **TVGL**: 1.878408

Both methods exhibit strong Sharpe Ratios, with TVGL slightly outperforming Inverse Covariance.

3. Maximum Drawdown Maximum Drawdown assesses the largest peak-to-trough decline, reflecting downside risk.

- **Inverse Covariance**: 0.080481 (8.0481%)
- **TVGL**: 0.098915 (9.8915%)

While TVGL achieved higher returns, it also experienced a slightly larger drawdown compared to Inverse Covariance.

4.4 Visualization

Figure 7 illustrates the cumulative returns over time for both methods.

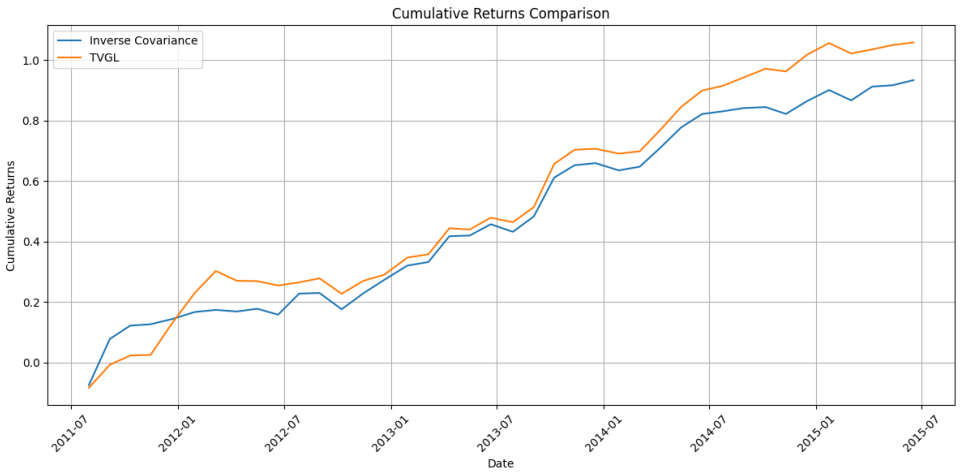

Figure 7: Cumulative Returns Comparison: Inverse Covariance vs. TVGL

4.5 Discussion

The comparative analysis reveals that the TVGL method outperforms the traditional Inverse Covariance approach in terms of cumulative returns and Sharpe Ratio. This suggests that TVGL's ability to capture temporal dependencies and structural changes in the asset relationships leads to more effective portfolio optimization. However, TVGL also incurs a slightly higher Maximum Drawdown, indicating increased exposure to downside risk during certain periods.

4.5.1 Possible Reasons for TVGL's Superior Performance

- **Temporal Dynamics**: TVGL accounts for time-varying relationships among assets, allowing the model to adapt to changing market conditions more effectively than the static Inverse Covariance method.
- **Sparsity and Smoothness**: The regularization parameters λ and β in TVGL enforce sparsity and smooth transitions between precision matrices, enhancing the stability and interpretability of the model.
- **Overfitting Prevention**: By incorporating regularization, TVGL mitigates the risk of overfitting, leading to more robust out-of-sample performance.

4.5.2 Considerations and Future Work

While TVGL demonstrates superior performance, the slightly higher drawdown warrants further investigation. Future work could explore:

- **Hyperparameter Optimization**: Fine-tuning λ and β to balance return maximization and drawdown minimization.
- **Inclusion of Transaction Costs**: Including transaction costs could provide a more realistic assessment of the strategy's performance.
- **Expanded Asset Universe**: Testing the strategy on a broader set of assets to evaluate scalability and robustness.
- **Alternative Portfolio Optimization Techniques**: Exploring mean-variance optimization or incorporating expected returns could enhance performance.