ChatGPT

Topological Allocator Regime Strategy – A Comprehensive Technical Report

1. Theoretical Framework & Financial Logic

Persistent Homology as an Early Warning System in Finance

Topological Data Analysis (TDA) and Market Crashes: In finance, **persistent homology** provides a way to detect subtle structural changes in market data that often precede major regime shifts or crashes. By constructing *point cloud* representations of financial time series and computing their homological features, we capture the "shape" of market dynamics beyond what linear indicators (like volatility or correlations) can reveal [1] [2]. Specifically, **Betti numbers** (β<sub>0</sub>, β<sub>1</sub>, etc.) measure the count of topological features in the data manifold: β<sub>0</sub> counts connected components (clusters of co-moving assets), and β<sub>1</sub> counts loops (cyclical relationships) [3] [4]. In stable market regimes, we often observe one dominant cluster of assets moving together (β<sub>0</sub> ≈ 1) and few persistent loops. However, as instability grows, the market's high-dimensional structure **fragments** into multiple clusters (β<sub>0</sub> increases) or exhibits complex loops, reflecting emergent co-movement patterns [1] [2]. This **collapse or formation of high-dimensional structures** is a hallmark of phase transitions in markets (analogous to physical phase changes), and TDA can detect it early. For example, an increase in the number of persistent loops or disconnected clusters can signal that diversification is breaking down and a crash may be looming [5] [6].

Persistence Landscapes and Norms: Rather than tracking raw Betti numbers alone, TDA often summarizes persistence data in **persistence diagrams** or **persistence landscapes**, which record the "lifetimes" of topological features across scales [7] [8]. A long-lived topological feature (e.g. a loop that persists over many scales) indicates a robust structural pattern in the data. The *persistence landscape* provides a functional summary of these lifetimes, and one can compute an $L^p$ **norm** of the landscape as a single indicator of topological complexity [8]. Crucially, research has shown that such persistence norms can serve as early warning signals for crises. For instance, the $L^1$-norm of the persistence landscape (essentially the total "length" of all persistence bars) tends to **rise in advance of major market crashes** [9]. In one study, the $L^1$ persistence norm began increasing well before the Dot-Com and 2008 crashes, providing a lead time on the order of weeks [10]. Intuitively, this norm spikes when there is a mix of many short-lived features and a few long-lived features – a sign of the market transitioning from orderly behavior to a stressed, disordered state. Correspondingly, **"topological volatility"** (the variance of feature lifetimes) often surges before traditional volatility does, acting as a geometric stress indicator that can **precede spikes in market volatility** [11] [12]. In other words, TDA-based signals like Betti counts or persistence norms can detect the early formation of instabilities (fragmentation of regime or emergent loops in correlations) **before** they fully materialize as a crash [2] [5].

Phase Transitions and High-Dimensional Collapse: During an actual crisis, correlations between assets famously all go to 1 (the "all correlations go to one" effect), which topologically corresponds to the market becoming one giant cluster (β<sub>0</sub> dropping to 1) and the disappearance of distinct loops or voids.

This **collapse of dimensional structure** is captured by a sharp change in persistent homology features. For example, studies have noted that the average values of persistence landscape norms **decrease sharply when covariance spikes** (as in March 2020 COVID crash) [13]. In the run-up to such events, however, persistent homology can detect subtle oscillatory patterns or new groupings that warn of a "critical transition." One recent framework demonstrated that by mapping sliding windows of multivariate returns into point clouds and computing Vietoris–Rips persistence, one can generate an $L^p$ topological indicator that gave a **34-day lead time** on average for major U.S. equity index crises [14] [15]. These findings reinforce that **persistent homology (via β<sub>0</sub>, β<sub>1</sub> and persistence landscapes) provides early warning signals** of phase transitions in financial markets by tracking the geometry of data in ways traditional time-series methods cannot [1] [6].

Topological Stability Index (TSI) – Measuring Structural Breaks

Definition of TSI: The **Topological Stability Index (TSI)** is a concise metric designed to quantify the degree of structural (in)stability in financial data using persistent homology. In practical terms, TSI is defined as the **variance of the persistence lifetimes** of topological features (usually considering both 0-dimensional and 1-dimensional features) in a given rolling window [16]. Formally, if we compute a persistence diagram for a time window's data (whether it's derived from a correlation network or an embedded point cloud of returns), and collect the lifetimes $\ell_i$ of all features (birth-to-death intervals), then:

- **TSI = Var(${\ell_i}$)**, the variance of those lifetimes [16].

This simple formula captures how heterogeneous the topological features' durations are. **Low TSI** means most features have similar short lifetimes – indicating a stable, homogeneous structure or simply random noise with no prominent features [17]. **High TSI** means a mix of some long-lived features and many short-lived ones, indicating a **heterogeneous or metastable structure** where the market is transitioning between order and chaos [18]. In economic terms, a **rising TSI** signifies periods of **structural reorganization** – for example, increasing stress in financial networks, sectors that were unified breaking into sub-groups, or new cyclic relationships emerging [19] [20]. This often corresponds to regime shifts: for instance, just before a crash or during a turmoil, the correlation matrix's structure becomes erratic (some correlations remain very strong while others decouple), yielding a high variance in persistence lifetimes.

TSI vs Linear Measures: TSI offers a **geometric perspective on market stability** that often outperforms traditional linear measures at detecting structural breaks. Traditional risk metrics (volatility, correlations, PCA eigenvalues, etc.) look at second-moment or linear relationships and might miss complex reconfigurations until they are extreme. In contrast, TSI monitors the "shape" of the correlation matrix or return manifold. Because it is built from **topological lifetimes rather than raw returns**, TSI can flag instabilities earlier. Notably, TSI is analogous to variance **but in the space of topological features rather than price returns** [21]. For example, an upward spike in TSI can reveal a breakdown in diversification or a sector decoupling **before** it's evident in rising price volatility [22]. One study emphasizes that an increase in TSI often **precedes visible market turbulence**, catching diversification failure ahead of time [22]. In effect, TSI extends risk management to a **structural level**: while volatility measures dispersion of returns, **TSI measures dispersion of correlation structure** [21]. Empirical evidence backs this up – cases have been documented where TSI began climbing in advance of major volatility spikes or correlation breakdowns, serving as an early warning signal of an oncoming regime change [11] [5]. Figure 22 in one reference illustrates this concept: as a calm market (features all short-lived, low TSI) transitions to a turbulent market (some features persist longer, indicating nascent subgroup formations), the TSI time-series trends upward **ahead of the realized crisis** [23] [24]. Because it is normalized and scale-agnostic, the TSI can be tracked just like a volatility index – a rising TSI is analogous to rising volatility-of-volatility or entropy, signaling the market's correlation structure is destabilizing [25] [22]. In summary, the **Topological Stability Index quantifies structural breaks in correlation matrices more sensitively than linear models**: it captures the **onset of correlation network fragmentation** or reorganization that linear correlation or PCA analyses might only catch in hindsight.

Hierarchical Risk Parity (HRP) – Robust Allocation for Regime Shifts

HRP Algorithm Overview: Hierarchical Risk Parity (HRP) is an asset allocation algorithm that uses hierarchical clustering and recursive optimization to allocate capital in a more robust way than traditional mean-variance optimization (MVO). Developed by Marcos López de Prado (2016), HRP addresses the pitfalls of Markowitz's MVO – namely instability to input error, concentration in few assets, and poor out-of-sample performance [26] [27]. The core idea is to **avoid inverting the covariance matrix or relying on unstable estimates of return**, and instead harness the structure of asset correlations. HRP works in three main steps [28]:

1. **Tree Clustering:** Compute a distance matrix between assets based on correlation (e.g. $d_{ij} = \sqrt{\frac{1}{2}(1-\rho_{ij})}$) [29]. Then perform hierarchical clustering (e.g. single-linkage or similar) to produce a dendrogram (binary tree) of assets [30] [31]. Assets that move similarly will be linked in the same cluster/subtree. This **tree structure** reveals natural groupings (such as sectors or asset classes) without any look-ahead – it's entirely based on historical correlations.

2. **Quasi-Diagonalization:** Reorder the covariance (or correlation) matrix according to the clustering order [32]. Essentially, we **permute** the matrix so that similar assets (those clustered together) are next to each other, which makes the covariance matrix **approximately block-diagonal** [32] [33]. López de Prado calls this "matrix seriation" or quasi-diagonalization. Unlike PCA, this does **not** change the basis or require eigen-decomposition; it simply **shuffles rows/columns** to align assets by cluster [34]. The result is that we isolate clusters of assets – groups that are highly correlated among themselves but relatively uncorrelated with others will appear as blocks on the diagonal of the reordered matrix [35]. This step is crucial because it localizes portfolio decisions: errors in estimating one block (cluster) won't immediately spread to unrelated assets.

3. **Recursive Bisection Allocation:** Finally, HRP allocates weights using a top-down recursive process on the clustered tree [36]. Starting with the two broadest clusters from the dendrogram (the final split of the tree), HRP splits total portfolio capital between these clusters in inverse proportion to their aggregated risk (often measured by the sum of variances within each cluster) [37]. Then it recurses into each cluster: for each parent node, split its allocated capital between its two sub-clusters according to inverse variance of those sub-clusters, and so on until reaching individual assets [38]. Within the smallest cluster (leaf level), if only one asset remains, it simply receives the allocated weight; if multiple, the process continues until fully allocated. This **recursive bisection** ensures **risk is spread out** roughly equally across clusters – effectively achieving a risk parity across the hierarchy of correlated assets, rather than across individual assets directly [39]. Assets only "compete" with similar assets for weight, avoiding the problem in MVO where a low-correlation outlier asset can dominate the portfolio weight because of optimization artifacts [40] [41]. HRP's output is a set of weights that are **better diversified and more stable** out-of-sample than traditional methods [27] [42].

Why HRP Handles Regime Shifts: A key advantage of HRP is its **robustness to changes in the covariance structure**, which is especially relevant during regime shifts. Traditional MVO portfolios can **swing wildly** or become invalid when the market regime changes (e.g., a sudden jump in correlations can make the previously optimal weights dangerously concentrated or even infeasible if the covariance matrix becomes singular). HRP, by contrast, inherently adapts to the correlation structure at each rebalancing via clustering. As soon as the TDA regime detection signals a shift, one can rebuild the hierarchical tree on recent data; **HRP will automatically re-cluster assets according to the new regime's correlations** and reallocate capital accordingly. This means that if previously uncorrelated assets become tightly correlated in a crisis, HRP will group them and ensure the combined cluster doesn't overwhelm the portfolio risk. Conversely, if a new safe-haven cluster emerges, HRP can isolate its risk. Importantly, HRP does **not require an invertible covariance matrix**, so in extreme regimes where many assets become perfectly correlated (covariance matrix nearly singular), HRP still functions, whereas traditional MVO would break down [43] [44]. Studies have shown HRP yields **lower out-of-sample volatility than even minimum-variance optimization**, highlighting its stability [45].

In the context of our Topological Allocator strategy, TDA will **identify the regime (e.g. calm vs stressed)** and HRP will ensure that within each regime the allocation is robust. For instance, when TDA signals a **Risk-Off regime** (market instability rising), one might reduce exposure to high-volatility clusters or tilt towards defensive asset clusters. HRP will facilitate this by naturally reducing weight on the volatile cluster if its internal variance is high. Conversely, in a **Risk-On regime** (stable structure), HRP spreads risk across all clusters, allowing fuller exposure to risk assets while still guarding against any one sector dominating. Moreover, HRP's **dynamic clustering** property means it *updates the portfolio structure as correlations shift* [46]. This adaptability is crucial during regime transitions identified by TDA. For example, if TDA's topological indicators show that previously unified market structure is fragmenting into multiple clusters, HRP will pick up that fragmentation in its clustering step and allocate to each new cluster separately, thereby **limiting concentration risk**. In volatile periods, this approach has been shown to yield better risk-adjusted returns and shallower drawdowns than static methods [46] [47]. In summary, **HRP overcomes the instability of mean-variance optimization via clustering and quasi-diagonalization**: it localizes estimation errors and naturally **adapts to regime shifts** by reallocating capital in sync with the evolving correlation topology [46] [48]. During calm regimes it behaves similar to traditional risk parity, while during stressed regimes it "reacts" to the new correlation blocks, maintaining portfolio resilience even as the market's structure changes.

2. Developer Implementation Guide

Data Pipeline and Preprocessing

Universe & Data Frequency: First, define the universe of assets (e.g. equities across sectors, bonds, commodities – excluding crypto as per instructions) and the data frequency. The strategy can be applied on **intraday, daily, or weekly** data, with appropriate parameter adjustments. For example, **intraday** regime detection might use high-frequency returns (e.g. 5-minute bars) and shorter rolling windows to capture swift micro-regime changes, whereas **daily** or **weekly** data would use longer windows to capture macro regime shifts. In all cases, we begin by **collecting historical price data** for each asset and converting it to **log-returns** (which stabilizes variance and allows additive time-series analysis). Log-returns should be cleaned for missing data and outliers (e.g. winsorize or cap extreme values) to ensure numerical stability for TDA calculations.

Sliding Window Preparation: Choose a window length W (number of time periods in each analysis window) and a stride (how often to update the analysis). For daily data, a common choice might be a 60-day or 90-day rolling window (roughly 3–4 months) that balances having enough data to compute topology while being short enough to detect changes. For intraday, W could be in hours or days of high-frequency points. We will generate a **rolling window dataset**: at each rebalancing time t, take the past W periods of log-returns for all assets as the current window. Each window will be treated as a separate point cloud for TDA and a separate sample for regime classification.

Takens Embedding for Point Clouds: To apply persistent homology, we need to represent time-series data as a *point cloud in a metric space*. There are two primary approaches (which are not mutually exclusive):

- *Approach A: Multivariate Return Point Cloud.* Treat each day (or period) in the window as a point in $\mathbb{R}^N$, where N is the number of assets. For example, if we have 10 assets and a 60-day window, we have a point cloud of 60 points in $\mathbb{R}^{10}$ (each point is the vector of 10 returns on a given day). We can use the Euclidean distance between points (or a Mahalanobis distance) to feed into persistent homology. This approach captures the **distribution of returns** in that window – e.g. whether returns cluster or form loops over time.

- *Approach B: Univariate Takens Embedding.* For each asset (or a representative index), we can form a **delay embedding** using Takens' theorem. For instance, to analyze an index's behavior, embed its 1-D time series from the window into an m-dimensional space by taking lagged vectors: $X_t = [r_t, r_{t-\tau}, r_{t-2\tau}, \dots, r_{t-(m-1)\tau}]$ (with appropriate choices of embedding dimension m and delay τ). The collection of such vectors over the window forms a point cloud in $\mathbb{R}^m$ that represents the state-space trajectory of that asset. Takens embedding can reveal cyclic attractors or strange geometry in a single time series that might signal a bubble or regime change. For example, a log-periodic oscillation in prices (which often precedes crashes) would appear as a loop in the embedded space, yielding a persistent H_1 cycle (β_1) [49] [50].

In practice, our **Topological Allocator** can utilize Approach A for capturing **cross-sectional structure** (relationships among multiple assets) and Approach B for capturing **temporal patterns** in individual series. Indeed, researchers have combined these: e.g. mapping sliding windows of multivariate returns to point clouds (Approach A), then using persistence landscapes or norms as features [51], or computing a "turbulence index" via Takens embedding of indices (Approach B) to detect structural breaks [52]. For this strategy, we focus on the **correlation structure approach**, which aligns with Approach A and the construction of TSI. Thus, for each window, we will also compute the **correlation matrix** of asset returns and derive a distance matrix $D_{ij} = \sqrt{\frac{1}{2}(1-\rho_{ij})}$ [29]. This distance matrix can be directly fed into a Vietoris–Rips complex algorithm (it defines a metric space of assets).

(Parameter tuning): If using Takens embedding, use methods like False Nearest Neighbors (FNN) to choose embedding dimension m and delay τ that sufficiently unfold the dynamics without excessive redundancy [53]. For example, an FNN analysis might show that $m=3$ and $\tau=5$ days minimize false neighbors for an index time series [54]. These parameters can be kept constant in backtests. For approach A, the main parameter is window size W and perhaps a normalization (we might z-normalize each asset's returns in the window to focus purely on correlation shape rather than scale differences [55]). Ensure each window's data is normalized appropriately (e.g. zero-mean, unit-variance) so that distances aren't dominated by scale.

Topological Feature Extraction (Persistent Homology)

Vietoris–Rips Complex Computation: For each rolling window's point cloud or distance matrix, compute the **persistent homology** up to dimension $p=1$ (that is, track connected components and 1-dimensional loops). We will use a Vietoris–Rips complex, which is standard for point cloud data: essentially, imagine gradually "connecting" points that are within an $\epsilon$ distance – at $\epsilon=0$ each point is isolated (β<sub>0</sub> = number of points), and as $\epsilon$ grows, clusters form (β<sub>0</sub> decreases), and eventually loops may form and then fill in (β<sub>1</sub> rises then falls). The output of this computation is a **persistence diagram**: a set of points $(b_i, d_i)$ for each topological feature, marking the "birth" and "death" $\epsilon$ values at which the feature appears and disappears. We obtain two sets of features: $H_0$ features (connected components) and $H_1$ features (loops).

Using Python Libraries: The Python ecosystem has well-developed libraries for these tasks:

- `giotto-tda`: A high-level library that provides transformers for TDA. We can use `giotto-tda` to perform Takens embedding (it has time series embedding tools), to compute persistence diagrams, and even to transform diagrams into vectorized features like persistence landscapes or entropies [2] [8]. For example, `giotto-tda` has a `VietorisRipsPersistence` class which can take either a point cloud or a precomputed distance matrix and output persistence diagrams for specified homology dimensions. It also has a `PersistenceLandscape` or `Amplitude` transformer that can compute $L^p$ norms of diagrams directly (via persistence landscapes or other summaries). Using `giotto-tda` pipelines, one could chain: embedding -> persistence -> landscape -> norm, in a scikit-learn style pipeline.

- `ripser`: A lower-level library (and also used under the hood by giotto) for fast computation of persistence diagrams, especially for Vietoris–Rips complexes. `ripser` can directly take a distance matrix or point cloud (as a NumPy array) and return persistence pairs. It's extremely fast due to optimized algorithms (especially important if using intraday data with many points). We might use `ripser` if we need fine control or to integrate into a custom loop (e.g., using `ripser.ripser(point_cloud, maxdim=1)` to get diagrams).

- `Persim` **or** `scikit-tda`: This includes utility functions to visualize diagrams or compute persistence landscapes if needed. However, `giotto-tda` now covers most functionality in one place.

For our implementation, we could do something like:

```python
from gtda.time_series import TakensEmbedding
from gtda.homology import VietorisRipsPersistence
from gtda.diagrams import Scaler, Amplitude

# Example pipeline elements:
embedder = TakensEmbedding(parameters...)       # if using Approach B for a
single series
VR = VietorisRipsPersistence(homology_dimensions=[0,1])
scaler = Scaler()  # normalize diagram lifetimes if needed
```

```
amplitude = Amplitude(metric='landscape', order=1)  # computes L^1 norm of
landscape
```

We would apply these to each window's data. For Approach A (multivariate returns), we might skip `TakensEmbedding` and directly feed the correlation distance matrix into `VietorisRipsPersistence` (which accepts a `metric="precomputed"` distance matrix). The output from `VR` is a list of persistence diagrams (one per homology dimension). We can then compute **features** from these diagrams: candidates include **total persistence** (sum of lifetimes), **number of long bars**, **entropy**, or as asked, the **L^p norm of the persistence landscape**. The L^p norm is a robust single number that increases with both the number and longevity of topological features [8]. In fact, the **Topological Stability Index (TSI)** we described is essentially related to the variance of lifetimes, which is directly connected to the spread of points in the persistence diagram [16] [22]. We can compute TSI by taking the lifetimes from the diagram and calculating their variance in Python.

Interpreting Topological Features: Each window thus yields one or more numeric features summarizing the market's shape. For instance, one could produce: `TSI_t` (topological stability index for window ending at t), `Beta1_count_t` (the number of persistent loops above some threshold in that window), and `PersNorm_t` (the L^1 persistence norm of the landscape). These indicators will be the inputs to our **regime decision logic**. Typically, we expect that **high values of TSI or persistence norm correspond to "Risk-Off" (unstable regime)**, while **low values correspond to "Risk-On" (stable regime)** [22] [12]. In backtesting, one would confirm this alignment by checking known crisis periods: e.g., does the 2008 crisis show a spike in TSI or persistence norm? Prior research and our earlier discussion indicate yes – topological turbulence indexes spiked during 2008 and 2020, and even gave early warnings for some events [15] [56].

Regime Switching Logic (Dynamic Reallocation using TDA + HRP)

With the topological features computed for each window, we implement a **Switcher Regime** mechanism that dynamically adjusts the portfolio's asset allocation between a risk-on posture and a risk-off posture. The regime decision can be made by comparing the persistence-based indicators to thresholds or by more sophisticated pattern recognition, but a straightforward approach is often effective:

Defining Risk-On vs Risk-Off Portfolios: Determine what "Risk-On" and "Risk-Off" mean for your strategy. For example, in a multi-asset portfolio: - **Risk-On** could mean a higher allocation to equities, high-yield credit, or cyclical assets. - **Risk-Off** could mean a tilt toward Treasuries, gold, defensive sectors, or simply a lower overall leverage (more cash).

In our strategy, we will maintain two sets of target weights: - `w_on` = weights for Risk-On regime (e.g., an HRP allocation across mostly risk assets). - `w_off` = weights for Risk-Off regime (e.g., an HRP allocation favoring defensive assets, or an HRP allocation across all assets but with an overall de-leveraging factor applied).

We use HRP to compute these weights given the latest data: - Whenever we decide to rebalance, we run the **HRP algorithm** on the current covariance matrix of asset returns (likely using the same window of data or a slightly longer one for stable covariance estimates). The `PyPortfolioOpt` library can compute HRP weights easily: for instance, using `HRPOpt` from `pypfopt.hierarchical_portfolio`. This will give a set of weights that are inherently diversified according to the current correlation structure [28]. - We might compute two HRP allocations: one on the full asset universe (for risk-on, assuming we want some of everything), and one on a subset or modified universe for risk-off. For example, `w_off` might be computed on a subset of safer assets or by imposing constraints (like max equity weight). Alternatively, a simpler method is to use the **same HRP weights** but scale down the total exposure in risk-off (i.e., go partially to cash or hedges).

Dynamic De-leveraging / Re-weighting Rule: Now, using the persistence norm or TSI indicator, we create a rule such as:

- If the chosen topological indicator exceeds a certain **upper threshold** (signaling abnormal structural instability), switch to **Risk-Off regime** allocation `w_off` (i.e., reduce risk). This might be triggered, for example, when TSI or $L^1$ norm breaches the 90th percentile of its historical distribution, or if it shows a sustained upward trend over several windows [9] [15]. We can incorporate a bit of hysteresis or smoothing: e.g., require the signal to be high for at least 2 consecutive periods or use a moving average of the indicator to avoid whipsaw. In the academic example we cited, they used a decision rule with run-length parameters to **suppress isolated spikes** and confirm regime shifts [57] [15]. We can do similarly – e.g., enter risk-off only if the indicator stays above threshold for a few days, and exit risk-off only after it falls below a lower threshold.

- If the indicator is below a **lower threshold** (signaling structural stability and normalcy), switch to **Risk-On regime** and apply `w_on`. In normal times, we want to participate in the market's gains fully, so when topology indicates a well-organized market (low TSI, few long-lived loops – basically one big happy cluster of assets moving steadily), we allocate aggressively to risk assets (still diversified by HRP to avoid idiosyncratic pitfalls).

- In intermediate cases, one can either define a neutral stance or continue in the previous regime until a clear signal emerges. Some implementations use a **buffer zone** between on/off thresholds to reduce flipping. For instance, if using z-scores: "risk-off if Z > 2, risk-on if Z < 1, otherwise hold previous allocation."

Pseudo-Code Outline:

```
# Precompute risk-on and risk-off HRP weights (or define process to get them)
w_on  = HRP_weights(asset_universe, cov_matrix)         # e.g., mostly equities
included
w_off = HRP_weights(defensive_universe, cov_matrix)     # e.g., tilt to bonds/
gold, or same universe with constraints)

current_regime = 'ON'
for each rebalance date t:
    # Compute TDA features for window ending at t
    diagram = VietorisRipsPersistence(window_data_t)
    TSI_t   = variance_of_persistence_lifetimes(diagram)
    L1norm_t = persistence_landscape_norm(diagram, p=1)

    # Regime decision based on threshold
    if L1norm_t > high_threshold and current_regime != 'OFF':
        current_regime = 'OFF'
        weights = w_off
    elif L1norm_t < low_threshold and current_regime != 'ON':
        current_regime = 'ON'
        weights = w_on

    execute_portfolio_rebalance(weights)
```

In this pseudocode, `high_threshold` and `low_threshold` could be predetermined based on historical analysis (e.g., a certain absolute level of the persistence norm that in backtesting corresponded to known crises, or perhaps using quantiles). The use of `current_regime` ensures we don't churn the portfolio unless there is an actual regime change signal. When switching to **Risk-Off**, if our design is to de-leverage, `w_off` might be implemented as, say, 50% allocated per HRP weights and 50% in cash or Treasury bills (meaning the portfolio volatility is reduced). If implementing a full **risk parity across assets** strategy, `w_off` could also mean including an inverse volatility target that's lower than in risk-on (essentially cutting exposure). The exact mechanics can be adjusted to the user's portfolio preferences.

Library Roles in Implementation Stack: To summarize the tools in the Python stack and their roles: - `pandas` **/NumPy**: Data handling, rolling windows, and basic stats on returns. - `giotto-tda` **/** `ripser`: Compute persistent homology (diagrams) and transformations like persistence landscapes. For example, giotto-tda can directly give us the L^1 norm of persistence landscapes via its `Amplitude` transformer [58] [59]. This greatly simplifies computing the persistence norm at each step. - `PyPortfolioOpt`: Compute the HRP weights. The library's `HRPOpt` class can take a covariance matrix and produce hierarchical risk parity allocations [60]. We simply supply the latest covariance (or correlation) matrix of asset returns. PyPortfolioOpt will handle clustering (it uses SciPy's linkage under the hood) and output a weight dictionary. - **Supporting libs**: `scikit-learn` (optional, if we integrate the pipeline in a CV framework), plotting libraries to visualize regime classification over time, etc. For performance on intraday data, one might consider using GPU acceleration for TDA (not widely available yet) or at least ensure the code is vectorized (giotto-tda is C++ optimized).

Example Workflow for the Switcher Regime

To make this concrete, imagine we are running the system daily: - Every day after market close, take the last 60 trading days of returns. - Use `giotto-tda` to get the persistence diagram for H_0 and H_1. Compute the TSI (variance of lifetimes) or L^1 persistence norm from that diagram. - If the norm is, say, above a threshold corresponding to the 95th percentile of its 1-year history, we flag a **Risk-Off** regime starting tomorrow. The system might, for example, rebalance to 30% equity HRP portfolio / 70% bonds HRP portfolio (effectively a weighted combination that's more defensive), or simply reduce equity weights within the HRP solution [47]. If we had already been in risk-off and the indicator remains high, we stay put. - If the norm falls back below a calmer threshold (e.g. 50th percentile), we switch back to **Risk-On**, deploying full risk allocation per HRP on the broad asset set. - If the indicator is in between, we do nothing, keeping current weights.

Over time, this dynamic switching should **de-leverage during topologically unstable periods** (thus avoiding large drawdowns) and **re-risk during stable periods** (capturing upside). The expectation from theory is that this will improve the portfolio's **Sharpe ratio and drawdown profile** by sidestepping the worst crashes without completely timing the market (we are using robust structural signals rather than price predictions). Crucially, because HRP is used for allocation, even within each regime the portfolio is constructed to be resilient (no concentrated bets that could be particularly wrong if we slightly mistime a regime).

3. Validation and Backtesting

Walk-Forward Backtesting to Avoid Look-Ahead Bias

When developing a strategy that involves regime detection (especially one as complex as TDA-based detection), it is essential to use **walk-forward methodology** for backtesting. This ensures **strict temporal causality**: at each point in the backtest, our algorithm only has information from the past and none from the future [61] [62]. A common pitfall in academic studies is that they delineate regimes in hindsight or use the full sample to train a clustering model, leading to overly optimistic results. In practice, **regimes often "change retrospectively" when new data arrives**, meaning a model that looks clean on historical clusters can repaint history as conditions evolve [63] [64]. To combat this:

- **Rolling Origin Evaluation:** We partition the historical data into sequential periods. For example, use data up to 2015 to calibrate initial thresholds for TSI/persistence norm (or even train a classifier if we were using one to map topology to regimes), then test the strategy from 2016–2018. Then roll forward: recalibrate thresholds or model up to 2018, test 2019–2020, and so on. This "walk-forward" simulation respects the timeline, updating the strategy's parameters as a real trader would over time.

- **Online Clustering vs Offline Clustering:** If we use any clustering or classification on persistence features to define regimes (for instance, one could cluster the persistence-norm time series itself to let the data decide what constitutes high vs low instability), that clustering must be done in an expanding window manner. For simplicity, our strategy used fixed thresholds, but even those can be treated as parameters to optimize. We should be careful **not to peek into the future** when setting these. A robust approach is to use **only pre-2008 data** to determine that a certain TSI level corresponds to risk-off, and then see if it correctly anticipates 2008 in an out-of-sample test. The **Medium article by Ansique (2025)** underscores that many regime detection methods fail in live trading because their backtests allowed regimes to be defined with future data, causing **performance to degrade in real time** [65] .

- **Walk-Forward on HRP Allocation:** The HRP itself can be part of the backtest loop. At each rebalancing date in the backtest, use only historical returns up to that date to compute the hierarchical clustering and weights. This ensures we're not using "future correlation" information. Typically, this is fine because HRP inherently is backward-looking (it uses the latest covariance matrix). But one must ensure to **rebalance HRP weights only on the schedule** one would in reality (e.g., don't use tomorrow's covariance today). In code, one would step day by day (or period by period), update the regime signal and portfolio, then advance.

- **Transaction Costs & Latency:** Incorporate assumptions for slippage, transaction costs, and the fact that signals come with a slight delay (e.g., if using daily bars, you'd execute at next day's open or close). The topological computations are fast enough that in practice one could generate the signal shortly after close and trade at next open. But in an intraday setting, one might do the analysis with a slight lag (e.g., use up to 11:00 AM data to decide if by noon you de-risk).

Performance Evaluation and Verification

To validate the strategy, perform a **walk-forward backtest** from, say, 2000 through 2025 (daily data) or a few years if intraday. Key metrics to evaluate: CAGR, volatility, Sharpe ratio, Max Drawdown, and Calmar ratio, comparing the Topological Allocator vs a static HRP or vs a classical 60/40 or risk parity. We expect to see comparable returns with reduced drawdowns during crisis periods. For example, one should verify that before the 2008 crash, the strategy indeed shifted to risk-off (de-risking the portfolio in late 2007 or 2008 Q1), and similarly for the 2020 COVID crash (ideally de-risking in Feb 2020) – the research cited earlier did show successful early warnings for these events [15]. Check also that in prolonged stable rallies (2017, 2019, etc.), the strategy isn't too under-invested (false alarms should be minimal). The MDPI 2025 study showed that by using a rigorous topology-based signal with causal detection rules, they achieved **fewer false alarms than volatility-based triggers at comparable recall levels** [66]. We aim for the same: the walk-forward test should show that the topology-driven switch avoids most major crashes while only rarely missing out on gains due to false risk-off signals.

Robustness Checks: Conduct sensitivity analysis on the window size W, the persistence norm threshold, and the HRP rebalancing frequency. Perhaps test weekly vs monthly rebalancing, or using an L^2 norm instead of L^1 (some studies found both L^1 and L^2 norms of persistence useful [56]). The strategy should be reasonably robust – small changes might affect exact entry/exit timing but not the overall benefit. If the results are highly sensitive, one might refine the approach (e.g., combine topological indicators with a confirmatory traditional indicator like VIX or credit spreads to filter noise).

Finally, before actual trading, do an **out-of-sample forward test** (paper trade) for a while, and ensure operational aspects (data latency, etc.) are handled. The entire pipeline – from data ingestion to feature calculation to portfolio optimization – must run reliably and quickly. Libraries like `giotto-tda` and `PyPortfolioOpt` make it feasible to implement this pipeline in a research environment; moving to production might involve translating parts to a lower-level language or ensuring that the Python code can handle real-time data streams (for intraday, consider pre-computing as much as possible during off hours).

In conclusion, by rigorously backtesting with walk-forward analysis and respecting temporal integrity, we can confidently validate that the **Topological Allocator Regime strategy** offers a compelling blend of **early anomaly detection** (via TDA: persistence homology and TSI detecting regime shifts before they fully happen) and **robust adaptive allocation** (via HRP adjusting portfolio weights to the detected regime). This synergy is designed to protect portfolios during market phase transitions while still capitalizing on growth during stable periods – a modern, data-driven answer to the challenges of regime changes in financial markets.

Sources:

- Hobbelhagen & Diamantis (2024), *The Shape of Data: Topology Meets Analytics* – on TDA capturing hidden market geometry and the Topological Stability Index [12] [22].
- Guritanu *et al.* (2025), *Topological ML for Financial Crisis Detection* – early warning via persistence landscape norms [2] [57].

- López de Prado (2016), *Hierarchical Risk Parity* – algorithm and advantages of HRP over Markowitz [28] [43] .
- LuxAlgo Research (2023), *HRP Rebalancing Methods* – notes on HRP's adaptability to shifting market dynamics [46] [47] .
- Ansique (2025), *Walk-Forward Market Regime Detection* – emphasis on avoiding look-ahead bias in regime identification [67] .

---

[1] [2] [3] [4] [6] [8] [14] [15] [49] [51] [52] [56] [57] [61] [62] [66] Topological Machine Learning for Financial Crisis Detection: Early Warning Signals from Persistent Homology
https://www.mdpi.com/2073-431X/14/10/408

[5] [7] [11] [12] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [53] [54] [55] The Shape of Data: Topology Meets Analytics A Practical Introduction to Topological Analytics and the Stability Index (TSI) in Business
https://arxiv.org/html/2511.13503v1

[9] [10] [13] [50] Time-resolved topological data analysis of market instabilities | Request PDF
https://www.researchgate.net/publication/349343706_Time-resolved_topological_data_analysis_of_market_instabilities

[26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [43] [44] [45] Hierarchical Risk Parity - Wikipedia
https://en.wikipedia.org/wiki/Hierarchical_Risk_Parity

[39] [40] [41] [42] [46] [47] [48] Hierarchical Risk Parity Rebalancing Methods
https://www.luxalgo.com/blog/hierarchical-risk-parity-rebalancing-methods/

[58] [59] Computing L^p Norms (specificically L^1 norm) from persistence diagram · giotto-ai giotto-tda · Discussion #647 · GitHub
https://github.com/giotto-ai/giotto-tda/discussions/647

[60] Portfolio Optimization with Python: Hierarchical Risk Parity - Yang Wu
https://kenwuyang.com/posts/2024_10_20_portfolio_optimization_with_python_hierarchical_risk_parity/

[63] [64] [65] [67] Market Regime Detection | by Ánsique | Nov, 2025 | Medium
https://medium.com/@Ansique/walk-forward-market-regime-detection-a-production-ready-implementation-b9ea346f52a3