



Article

Particle Swarm Optimization with Stretching and Clustering for Asset Allocation

Julien Chevallier 

Economics Department (LED), Université Paris 8, 2 Avenue de la Liberté, 93526 Saint-Denis, France;
julien.chevallier04@univ-paris8.fr

Abstract

This paper develops a novel hybrid framework that integrates clustering-enhanced Particle Swarm Optimization (PSO) with stretching techniques to solve Markowitz's quadratic portfolio optimization problem. The proposed approach avoids local optima traps that plague traditional optimization methods, while the stretching function modifications enhance the algorithm's global search capabilities. The framework comprises four distinct algorithmic variants: a baseline SWARM PSO with stretching algorithm, and three clustering-enhanced extensions incorporating Hierarchical, K-means, and DBSCAN techniques. These clustering enhancements strategically group assets based on risk–return characteristics to improve portfolio diversification and risk management. Implementation in R enables comprehensive analysis of portfolio weight allocation patterns and diversification metrics across varying market structures. Empirical validation using daily price data from six major international stock market indices spanning January 2020 to December 2025 demonstrates the framework's generalization capability in constructing buy-and-hold investment portfolios. The results reveal significant market-specific algorithmic effectiveness, with K-means variants achieving competitive efficacy in Eurostoxx and Belgian markets, DBSCAN demonstrating strong effectiveness in Chinese equity markets, Hierarchical clustering showing robust results in Indian market conditions, and the baseline SWARM algorithm exhibiting relative efficiency in French and Danish indices. Performance evaluation encompasses comprehensive risk-adjusted metrics, including Portfolio Return, Volatility, Sharpe Ratio, Calmar Ratio, and Value at Risk, providing portfolio managers with an adaptive, market-responsive optimization toolkit.

Keywords: asset allocation; particle swarm optimization; stretching; clustering; portfolio performance



Received: 8 December 2025

Revised: 16 January 2026

Accepted: 28 January 2026

Published: 4 February 2026

Copyright: © 2026 by the author.

Licensee MDPI, Basel, Switzerland.

This article is an open access article

distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)

[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Asset managers often encounter significant challenges when using Markowitz's (1952) mean-variance optimization. This method can produce unstable portfolio weights that vary greatly from one period to the next and may result in algorithmic non-convergence, which underscores the complexities and unpredictabilities of real-world asset allocation, making the application of mean-variance optimization a nuanced task that requires careful consideration.

In this paper, I propose a new mean-variance Markowitz solver programmed in R, leveraging evolutionary strategies. To overcome key challenges in quadratic programming, particularly in avoiding local optima, I combine Particle Swarm Optimization (PSO) with stretching and clustering functions.

Among heuristic-based optimization methodologies, the Particle Swarm Optimizer is particularly appealing due to its biological inspiration, reflecting the dynamics of fish schools and bird flocks, as first introduced by [Eberhart and Kennedy \(1995\)](#). PSO is a method where solutions are represented as vectors in an n -dimensional space, known as particles. Each particle tracks its best-known position and the best overall found by the swarm, guiding its movement and velocity in iterations. The particles collaborate to explore the search space and find optimal solutions.

On the one hand, the “stretching” technique by [Parsopoulos et al. \(2001a\)](#) is incorporated into the Particle Swarm Optimizer (PSO) to overcome the issue of getting trapped in local minima, which can hinder its effectiveness in complex optimization tasks. The stretching technique uses a two-step transformation: first, it raises the objective function to reduce undesirable local minima, and then it stretches the neighborhood of the trapped point to convert a local minimum into a local maximum. This approach enhances the algorithm’s ability to escape local traps, improving its global optimization capabilities and making stretched PSO a more reliable tool for solving complex problems across various fields.

On the other hand, the clustering methodology represents another contribution. I apply various machine learning algorithms, including hierarchical clustering, K-means, and density-based clustering (DBSCAN). I extend PSO’s global search abilities to find optimal cluster centroids and apply function stretching to avoid local minima. By partitioning the search space or particle population, this original theoretical framework enhances both exploration and exploitation, leading to effective solutions for complex multi-modal optimization problems and unsupervised learning tasks.

The core contribution of this research to the field of financial optimization and portfolio management is the development of a novel algorithm that combines particle swarm optimization, stretching, and clustering techniques, implemented on the R platform. This algorithm will be rigorously tested using historical stock market data spanning six global indices from January 2000 to December 2025. I evaluate the performance of a buy-and-hold portfolio using various metrics, such as the Sharpe and Calmar ratios.

The remainder of the paper is structured as follows. Section 2 contains a literature review of the latest developments in particle swarm optimization. Section 3 details the models with stretching and with clustering. Section 4 contains an empirical application dedicated to asset allocation. Section 5 concludes.

2. Theoretical Contributions to Particle Swarm Optimization: A Literature Review

The theoretical basis of PSO has evolved substantially since its introduction. [Sengupta et al. \(2018\)](#) surveyed its development, hybridization strategies, and convergence analysis, while [Freitas et al. \(2020\)](#) emphasized enhancements such as inertia weight and constriction factor to address stagnation in convergence. A comprehensive and useful review of meta-heuristics for portfolio optimization can be found in [Erwin and Engelbrecht \(2023\)](#).

Dynamic parameter adaptation plays a central role in boosting PSO’s performance. [Zhan et al. \(2009\)](#) proposed Adaptive PSO, enabling real-time detection of swarm states (e.g., exploration or convergence) to tune inertia and acceleration parameters accordingly. [Yao et al. \(2024\)](#) extended this by integrating adaptive inertia weights, reverse learning, Cauchy mutation, and Hooke–Jeeves local search, jointly enhancing convergence and escape from local optima.

To balance convergence and diversity, several structural PSO variants have emerged. [Lin et al. \(2025\)](#) segmented the swarm into subpopulations with customized learning rates and adaptive tuning, enabling local and global cooperation. [Jiang et al. \(2025\)](#) added

chaotic initialization, adaptive inertia, multi-subpopulation coordination, and mutation, collectively preventing premature convergence in complex landscapes.

Multimodal and multi-objective tasks have spurred efforts to enhance PSO's ability to capture multiple optima. Passaro and Starita (2008) combined PSO with dynamic k -means clustering to identify multiple optima in multimodal functions. Li et al. (2021) proposed a grid search-enhanced multi-population PSO combining clustering and spatial partitioning to maintain diverse Pareto sets and improve solution distribution.

In unsupervised learning, PSO has supported robust clustering. Huang (2011) paired PSO with Rough Set Theory to dynamically select the number of clusters, improving classifier accuracy. Zhang and Liu (2023) combined PSO with cloud theory and information entropy to boost feature selection and clustering on high-dimensional, imbalanced data. Verma et al. (2021) integrated fuzzy C-means with PSO for improved brain image segmentation, leveraging PSO's global search to overcome local minima in medical data classification tasks. Hayashida et al. (2025) introduce ELPSO-C, an enhanced leader particle swarm optimization variant that uses agglomerative clustering to detect and control dimension-wise diversity, selectively reintroducing variation in stagnating dimensions to improve convergence and search performance on high-dimensional optimization problems.

PSO applications in financial portfolio management have evolved from basic optimization to sophisticated frameworks addressing real-world constraints. Chen et al. (2021) demonstrated PSO's effectiveness in optimizing non-differentiable risk metrics, particularly the Sortino ratio, within large portfolio contexts where traditional gradient-based methods fail. Bulani et al. (2025) advanced this approach by integrating clustering methodologies with enhanced data preprocessing techniques, improving risk-adjusted returns across diverse asset classes and creating more robust optimization frameworks for complex financial environments. Lolic (2024) introduces two practical regularization enhancements to classical mean-variance optimization—one that constrains utility to reduce portfolio weight concentration, and another that resamples asset subsets—to produce more stable, diversified multi-asset portfolios with improved out-of-sample risk-adjusted returns relative to standard mean-variance optimization. Ntare et al. (2025) apply dynamic portfolio optimization and asset selection methods, including PSO, to examine diversification benefits and risk–return trade-offs when combining cryptocurrencies with highly correlated bank equities.

Domain-specific hybridizations expand PSO's reach. Aguiar Nascimento et al. (2022) proposed a two-phase optimizer for Full Waveform Inversion using modified PSO with k -means for exploration, followed by the Adaptive Nelder–Mead method for local refinement. Ananthi et al. (2025) developed a metaheuristic for dynamic data streams via lion optimization and an exponential PSO variant, enhancing centroid initialization for real-time clustering. Yuan et al. (2024) propose a multi-robot task allocation approach that synergistically combines a constrained K-means++ clustering (to group tasks according to robot capacity) with particle swarm optimization (to assign clusters to robots and optimize task execution order), demonstrating improved collaborative efficiency over other heuristic methods in both simulation and real robot experiments.

Extensions of PSO through alternative swarm models continue to diversify the algorithmic toolkit. Niu et al. (2025) introduced a multi-objective Sand Cat Swarm Optimization with adaptive clustering, refining crowding distance and improving neighborhood modeling in multimodal contexts, underscoring the integrative potential of non-PSO swarm paradigms.

Hybrid PSO architectures fuse complementary strategies for expanded functionality. Niknam and Amiri (2010) combined fuzzy adaptive PSO, ant colony optimization, and

k -means to reduce initialization sensitivity and improve clustering accuracy. [Abubaker et al. \(2015\)](#) introduced a hybrid of multi-objective PSO with simulated annealing, optimizing cluster validity metrics while mitigating stagnation. In another evolutionary framework, [Muteba Mwamba et al. \(2025\)](#) demonstrated the effectiveness of the Non-dominated Sorting Genetic Algorithm III (NSGA-III) over the traditional mean-variance optimization method for financial portfolio management.

3. Model

The modelling setup includes five main components: (i) [Ledoit and Wolf's \(2003\)](#) shrinkage estimator to estimate precisely the covariance matrix of assets; (ii) the particle swarm optimization with stretching for better search space exploration; (iii) hierarchical clustering for nested cluster structures; (iv) K-Means for data partitioning into fixed cluster numbers; and (v) Density-Based Spatial Clustering of Applications with Noise to identify clusters of different shapes and densities while managing outliers.

3.1. Quadratic Portfolio Optimization with Shrinkage Estimator

The integration of [Markowitz's \(1952\)](#) seminal portfolio optimization framework with [Ledoit and Wolf's \(2003\)](#) shrinkage estimator represents a sophisticated approach to addressing the fundamental challenges of modern portfolio theory in high-dimensional settings. This enhanced methodology combines the theoretical elegance of mean-variance optimization with advanced statistical techniques that mitigate the notorious instability of sample covariance matrices, particularly when the number of assets approaches or exceeds the number of observations.

3.1.1. [Markowitz's \(1952\)](#) Portfolio Optimization

The classical mean-variance optimization framework, introduced by [Markowitz \(1952\)](#), revolutionized portfolio management by formalizing the intuitive concept that investors seek to maximize expected returns while minimizing risk. This foundational approach treats portfolio construction as a quadratic optimization problem that explicitly balances the trade-off between expected return and variance, providing a mathematically rigorous foundation for rational investment decision-making.

Let $X = \{x_1, x_2, \dots, x_N\} \in \mathbb{R}^d$ be the set of raw individual stocks within an index, for which I compute the log-returns μ . The classical mean-variance optimization problem is formulated as:

$$\min_{\mathbf{w}} \quad \frac{1}{2} \mathbf{w}^\top \Sigma \mathbf{w} \quad (1)$$

$$\text{subject to} \quad \mathbf{w}^\top \mu = \mu_{\text{target}} \quad (2)$$

$$\mathbf{w}^\top \mathbf{1} = 1 \quad (3)$$

This optimization framework seeks to determine the optimal portfolio weights $\mathbf{w} \in \mathbb{R}^N$ that minimize portfolio variance while satisfying two fundamental constraints. The objective function $\frac{1}{2} \mathbf{w}^\top \Sigma \mathbf{w}$ represents the portfolio variance, where $\Sigma \in \mathbb{R}^{N \times N}$ is the covariance matrix of asset returns that captures both individual asset volatilities and cross-asset correlations. The quadratic form elegantly encapsulates the risk contribution of each asset and the diversification benefits arising from imperfect correlations between assets.

The first constraint, $\mathbf{w}^\top \mu = \mu_{\text{target}}$, ensures that the portfolio achieves a predetermined target expected return, where $\mu \in \mathbb{R}^N$ is the vector of expected returns for each asset and μ_{target} represents the investor's desired portfolio return. This constraint transforms the optimization problem from a simple variance minimization to a constrained optimization

that explicitly considers the return-risk trade-off. The second constraint, $\mathbf{w}^\top \mathbf{1} = 1$, where $\mathbf{1} \in \mathbb{R}^N$ is a vector of ones, ensures that the portfolio weights sum to unity, representing the requirement that the entire investment capital is allocated across the available assets.

3.1.2. Ledoit and Wolf's (2003) Shrinkage Estimator

The practical implementation of Markowitz's framework encounters significant challenges when dealing with sample covariance matrices, particularly in high-dimensional settings where the number of assets is large relative to the number of observations. Ledoit and Wolf (2003) addressed this fundamental limitation by developing a shrinkage estimator that systematically combines the sample covariance matrix with a structured target matrix, effectively reducing estimation error through a principled bias-variance trade-off.

The Ledoit and Wolf's (2003) shrinkage estimator of the covariance matrix Σ is defined as:

$$\Sigma_{\text{LW}} = \delta \cdot \mathbf{F} + (1 - \delta) \cdot \mathbf{S} \quad (4)$$

This shrinkage formulation represents a convex combination of two distinct covariance estimators, where $\mathbf{S} \in \mathbb{R}^{N \times N}$ is the sample covariance matrix computed from historical return data, and $\mathbf{F} \in \mathbb{R}^{N \times N}$ is the single-index model covariance matrix that serves as a structured target. The sample covariance matrix $\mathbf{S}$ is unbiased but exhibits high variance, particularly when the ratio of assets to observations is large, leading to unstable portfolio weights and poor out-of-sample performance. Conversely, the structured target $\mathbf{F}$ introduces bias but substantially reduces variance by imposing a parsimonious structure that reflects economic intuition about asset return relationships.

The shrinkage intensity parameter $\delta \in [0, 1]$ determines the optimal balance between bias and variance, with $\delta = 0$ corresponding to the sample covariance matrix and $\delta = 1$ representing complete reliance on the structured target. The genius of the Ledoit–Wolf approach lies in the analytical derivation of the optimal shrinkage intensity that minimizes the expected squared Frobenius norm of the estimation error, providing a theoretically grounded and computationally efficient solution to the covariance estimation problem.

The optimal shrinkage intensity δ^* is determined by minimizing the expected squared Frobenius norm of the estimation error:

$$\delta^* = \frac{\pi - \rho}{\gamma} \quad (5)$$

This optimal shrinkage intensity is computed using three fundamental components that capture different aspects of the estimation problem. The parameter π represents the asymptotic variance of the sample covariance matrix elements, quantifying the uncertainty inherent in the sample-based estimation:

$$\pi = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \text{AsyVar} \left[\sqrt{T} \cdot s_{ij} \right] \quad (6)$$

The parameter ρ measures the asymptotic covariance between the sample covariance matrix and the structured target, reflecting the extent to which the target matrix aligns with the sample-based estimates:

$$\rho = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \text{AsyCov} \left[\sqrt{T} \cdot s_{ij}, \sqrt{T} \cdot f_{ij} \right] \quad (7)$$

The parameter γ represents the squared Frobenius norm of the difference between the structured target and the true population covariance matrix, quantifying the bias introduced by the target structure:

$$\gamma = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N (f_{ij} - \sigma_{ij})^2 \quad (8)$$

These components involve the (i, j) -elements s_{ij} and f_{ij} of matrices $\mathbf{S}$ and $\mathbf{F}$, respectively, and σ_{ij} representing the (i, j) -element of the true population covariance matrix Σ . The parameter T denotes the number of observations used to compute the sample covariance matrix, while AsyVar and AsyCov represent asymptotic variance and covariance operators that capture the large-sample behavior of the estimators.

3.1.3. Enhanced Portfolio Optimization

The integration of Ledoit and Wolf's (2003) shrinkage estimator into Markowitz's (1952) framework creates a robust portfolio optimization approach that maintains the theoretical elegance of mean-variance optimization while addressing the practical limitations of sample covariance matrices. This enhanced formulation systematically improves portfolio performance by reducing estimation error and increasing the stability of optimal portfolio weights.

Incorporating the Ledoit and Wolf's (1952) shrinkage estimator into the Markowitz's (1952) framework yields:

$$\min_{\mathbf{w}} \quad \frac{1}{2} \mathbf{w}^\top \Sigma_{\text{LW}} \mathbf{w} \quad (9)$$

$$= \frac{1}{2} \mathbf{w}^\top (\delta^* \mathbf{F} + (1 - \delta^*) \mathbf{S}) \mathbf{w} \quad (10)$$

$$\text{subject to} \quad \mathbf{w}^\top \boldsymbol{\mu} = \mu_{\text{target}} \quad (11)$$

$$\mathbf{w}^\top \mathbf{1} = 1 \quad (12)$$

This enhanced formulation preserves the essential structure of the Markowitz optimization problem while substituting the problematic sample covariance matrix with the shrinkage estimator Σ_{LW} . The resulting optimization problem maintains the same constraints and objective function structure, ensuring compatibility with existing portfolio optimization algorithms and theoretical results, while significantly improving the numerical stability and out-of-sample performance of the resulting portfolios.

The shrinkage estimator effectively balances the bias-variance tradeoff inherent in covariance estimation: the structured target $\mathbf{F}$ provides regularization that reduces variance and improves numerical conditioning, while the sample covariance matrix $\mathbf{S}$ preserves data-specific covariance patterns and maintains a connection to the observed return dynamics. The closed-form solution for the optimal shrinkage intensity δ^* ensures consistency under large-dimensional asymptotics, providing theoretical guarantees for the estimator's performance as both the number of assets and observations increase.

This integration represents a significant advancement in portfolio optimization methodology, combining the foundational insights of modern portfolio theory with cutting-edge statistical techniques to address the practical challenges of high-dimensional portfolio construction. The resulting framework maintains the intuitive appeal and theoretical rigor of the Markowitz approach while providing substantially improved empirical performance in realistic investment settings.

3.2. PSO Dynamics with Stretching Function

3.2.1. Vanilla PSO

Particle swarm optimization operates through a population-based metaheuristic that simulates the collective behavior of bird flocking or fish schooling to solve optimization problems. The algorithm maintains a swarm of particles, where each particle represents a potential solution that moves through the search space by dynamically adjusting its position based on its own experience and the collective knowledge of the swarm.

According to [Eberhart and Kennedy \(1995\)](#), the fundamental mechanism governing particle movement consists of two sequential update equations that define how each particle i evolves from iteration k to iteration $k + 1$. The first equation updates the velocity vector, which determines the direction and magnitude of the particle's movement:

$$\mathbf{v}_i^{(k+1)} = w\mathbf{v}_i^{(k)} + c_1r_1(\mathbf{p}_i - \mathbf{x}_i^{(k)}) + c_2r_2(\mathbf{g} - \mathbf{x}_i^{(k)}) \quad (13)$$

This velocity update equation comprises three distinct components that balance exploration and exploitation. The first term, $w\mathbf{v}_i^{(k)}$, represents the inertia component, where the inertia weight w controls the influence of the particle's previous velocity direction. A higher inertia weight promotes global exploration by maintaining the particle's momentum, while a lower value encourages local exploitation by reducing the impact of previous movement patterns.

The second term, $c_1r_1(\mathbf{p}_i - \mathbf{x}_i^{(k)})$, constitutes the cognitive component that attracts each particle toward its personal best position $\mathbf{p}_i$. This term represents the particle's individual learning capability, where c_1 is the cognitive acceleration coefficient that determines the strength of attraction toward the particle's historical best performance. The random number $r_1 \in [0, 1]$ introduces stochastic variation that prevents deterministic behavior and maintains diversity in the search process.

The third term, $c_2r_2(\mathbf{g} - \mathbf{x}_i^{(k)})$, represents the social component that draws particles toward the global best position $\mathbf{g}$ discovered by the entire swarm. The social acceleration coefficient c_2 controls the intensity of this collective attraction, while the random number $r_2 \in [0, 1]$ ensures probabilistic movement toward the global optimum. This social learning mechanism enables information sharing among particles and facilitates convergence toward promising regions of the search space.

Following the velocity update, the second equation updates the particle's position by integrating the newly computed velocity:

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} + \mathbf{v}_i^{(k+1)} \quad (14)$$

This position update represents a simple Euler integration scheme that translates the particle's current location by the velocity vector, effectively moving the particle through the N -dimensional search space. The position vector $\mathbf{x}_i \in \mathbb{R}^N$ represents the candidate solution, which in portfolio optimization contexts corresponds to the portfolio weights across N assets.

The algorithm maintains memory structures for each particle, including the personal best position $\mathbf{p}_i \in \mathbb{R}^N$, which stores the best solution found by particle i throughout its search history, and the global best position $\mathbf{g} \in \mathbb{R}^N$, which represents the best solution discovered by any particle in the swarm. These memory components enable the algorithm to retain valuable information and guide future search directions based on accumulated experience.

The velocity vector $\mathbf{v}_i \in \mathbb{R}^N$ serves as the primary mechanism for exploration and exploitation, encoding both the direction and speed of particle movement. The interplay

between the inertia weight $w \in \mathbb{R}$, acceleration coefficients $c_1, c_2 \in \mathbb{R}$, and random numbers $r_1, r_2 \in [0, 1]$ creates a dynamic balance between diversification (exploring new regions) and intensification (exploiting promising areas), enabling the swarm to efficiently navigate complex optimization landscapes while avoiding premature convergence to suboptimal solutions.

3.2.2. Function Stretching Technique for PSO

The Function Stretching technique represents a sophisticated two-stage transformation method designed to help Particle Swarm Optimization (PSO) escape from local minima by systematically modifying the objective function landscape. This approach by Parsopoulos et al. (2001a), Parsopoulos et al. (2001b), Parsopoulos and Vrahatis (2002) and Parsopoulos and Vrahatis (2004) is particularly valuable because it addresses one of the most fundamental challenges in global optimization: the tendency of search algorithms to become trapped in suboptimal solutions.

Conceptual Foundation

The core philosophy behind Function Stretching lies in the strategic manipulation of the objective function's topology. Rather than attempting to modify the search algorithm itself, this technique transforms the problem landscape in a way that eliminates problematic local minima while preserving the global optimum. The transformation is applied immediately after PSO has converged to a local minimum $\bar{\mathbf{x}}$, essentially “reshaping” the function to create new pathways toward the global solution.

First-Stage Transformation: Elevation of Local Minima

The first transformation stage is designed to eliminate local minima that possess higher function values than the detected local minimum $\bar{\mathbf{x}}$. This is achieved through the following transformation:

$$G(\mathbf{x}) = f(\mathbf{x}) + \gamma_1 \|\mathbf{x} - \bar{\mathbf{x}}\| \cdot [\text{sign}(f(\mathbf{x}) - f(\bar{\mathbf{x}})) + 1]$$

The mathematical structure of this transformation is carefully constructed to achieve specific objectives. The term $\|\mathbf{x} - \bar{\mathbf{x}}\|$ represents the Euclidean distance from any point $\mathbf{x}$ to the detected local minimum $\bar{\mathbf{x}}$. This distance component ensures that the transformation's effect is spatially localized, with the strongest impact occurring near the problematic local minimum.

The sign function $\text{sign}(f(\mathbf{x}) - f(\bar{\mathbf{x}}))$ serves as a selective filter that determines which regions of the function landscape should be modified. When $f(\mathbf{x}) > f(\bar{\mathbf{x}})$, the sign function returns +1, and when combined with the +1 constant, produces a coefficient of 2. This means that points with higher function values than the local minimum receive the full strength of the transformation. Conversely, when $f(\mathbf{x}) < f(\bar{\mathbf{x}})$, the sign function returns −1, and the combined coefficient becomes 0, leaving these regions completely unaffected.

The parameter γ_1 controls the intensity of this elevation effect. A larger γ_1 value results in more dramatic elevation of the problematic regions, while a smaller value produces a gentler transformation. The choice of γ_1 must balance effectiveness in eliminating local minima with maintaining the overall function structure.

The geometric interpretation of this transformation is that it “lifts up” all regions of the function that have higher values than the detected local minimum, with the lifting effect being proportional to the distance from $\bar{\mathbf{x}}$. This creates a cone-like elevation centered at the local minimum, effectively making previously attractive local minima become less appealing to the optimization algorithm.

Second-Stage Transformation: Neighborhood Stretching

The second transformation stage focuses on the immediate neighborhood of the detected local minimum, further discouraging the algorithm from remaining in this region:

$$H(\mathbf{x}) = G(\mathbf{x}) + \gamma_2 [\text{sign}(f(\mathbf{x}) - f(\bar{\mathbf{x}})) + 1] \cdot \tanh(\mu(G(\mathbf{x}) - G(\bar{\mathbf{x}})))$$

This transformation builds upon the result of the first stage, $G(\mathbf{x})$, and applies an additional modification specifically designed to “stretch” the neighborhood around $\bar{\mathbf{x}}$ upward. The hyperbolic tangent function $\tanh(\cdot)$ is crucial here because it provides a smooth, bounded transformation that prevents the function from becoming unbounded while still creating significant local changes.

The argument to the $\tanh$ function, $\mu(G(\mathbf{x}) - G(\bar{\mathbf{x}}))$, represents the difference between the current function value (after first-stage transformation) and the transformed value at the local minimum. The parameter μ controls the steepness of this hyperbolic tangent, effectively determining how rapidly the stretching effect transitions from its minimum to maximum values.

The same selective mechanism from the first stage, $\text{sign}(f(\mathbf{x}) - f(\bar{\mathbf{x}})) + 1$, is employed here to ensure that only points with function values higher than the original local minimum are affected by this stretching operation. The parameter γ_2 scales the overall magnitude of the stretching effect.

The combined effect of this second transformation is to create a smooth, upward-stretching distortion in the neighborhood of the local minimum. This transformation converts the former local minimum into a local maximum, making it highly unattractive to the optimization algorithm while maintaining smooth transitions to surrounding regions.

Preservation of Global Structure

A critical property of both transformation stages is their selective nature regarding the global optimum and other local minima with lower function values. Since the transformations only affect regions where $f(\mathbf{x}) > f(\bar{\mathbf{x}})$, any local minimum with a lower function value than $\bar{\mathbf{x}}$ remains completely unaltered. This includes the global minimum, which by definition has the lowest function value of all points in the search space.

The sign function mechanism mathematically guarantees this preservation property. When $f(\mathbf{x}) \leq f(\bar{\mathbf{x}})$, the term $\text{sign}(f(\mathbf{x}) - f(\bar{\mathbf{x}})) + 1$ evaluates to either 0 or 1, but in the critical case where $f(\mathbf{x}) < f(\bar{\mathbf{x}})$, it becomes 0, completely nullifying the transformation effect. This ensures that the global optimum’s basin of attraction is preserved and potentially even enhanced relative to the eliminated local minima.

Implementation in Stretched PSO (SPSO)

The Function Stretching technique is integrated into PSO through a monitoring and transformation protocol. The algorithm initially applies standard PSO to the original objective function $f(\mathbf{x})$. A convergence detection mechanism continuously monitors the swarm’s progress, identifying when the algorithm has stagnated at a local minimum $\bar{\mathbf{x}}$.

Upon detection of local minimum convergence, the algorithm applies the two-stage transformation to create the new objective function $H(\mathbf{x})$. The PSO is then reinitialized with the same swarm configuration but now optimizes the transformed function. This process can be repeated multiple times if the algorithm encounters additional local minima, creating a sequence of progressively transformed functions that systematically eliminate problematic regions of the search space.

This Function Stretching approach represents a significant advancement in hybrid optimization strategies, combining the population-based search capabilities of PSO

with intelligent landscape modification techniques to achieve more reliable global optimization performance.

3.2.3. Comparison with Traditional Quadratic Solvers

Particle Swarm Optimization with Function Stretching offers practical advantages over conventional quadratic programming in both optimization robustness and computational efficiency (see Table A1 in the Appendix A to save space).

Namely, it enhances global optimization in non-convex portfolio settings by altering the objective landscape to suppress poor local minima while preserving global optima. This matters in real-world scenarios involving transaction costs and market frictions. Through an adaptive inertia weight $w(t)$, the swarm balances exploration and exploitation. Parallel evaluation of particles accelerates convergence in high-dimensional problems. Unlike traditional methods, PSO naturally handles various constraints without reformulation and remains stable even with noisy or ill-conditioned covariance input.

3.2.4. Pseudo-Code for PSO with Stretching

In Table A3 (see the Appendix A to save space), the algorithm begins by estimating a robust shrinkage covariance matrix Σ_{LW} using the Ledoit–Wolf procedure to mitigate instability in high dimensions. It generates diverse random portfolio weights $\mathbf{w}_i$ and evaluates portfolio risk via $f(\mathbf{w}) = \mathbf{w}^\top \Sigma_{LW} \mathbf{w}$. Function stretching transforms the landscape to avoid local optima, guiding the swarm toward the global solution. Particle positions and velocities are iteratively updated until convergence or stopping criteria are met. This hybrid method efficiently combines shrinkage estimation’s stability with PSO’s global search capacity to produce robust portfolios even with limited or noisy data.

3.3. Hierarchical Clustering of Assets as an Input to Markowitz Portfolio Optimization

3.3.1. Hierarchical Clustering of Assets

Hierarchical clustering transforms the enhanced covariance matrix into a tree structure that reveals natural groupings of financial assets. By using the Ledoit and Wolf’s (2003) shrinkage estimator’s correlations, it identifies assets with similar risk–return profiles, aiding in better portfolio construction and risk management.

Given a shrunk covariance matrix $\Sigma_{LW} = \delta^* \mathbf{F} + (1 - \delta^*) \mathbf{S}$ from Ledoit and Wolf (2003), I define a proximity matrix for hierarchical clustering:

$$\mathbf{D} = \{d_{ij}\}_{n \times n}, \quad d_{ij} = \sqrt{\sigma_{ii} + \sigma_{jj} - 2\sigma_{ij}} \quad (15)$$

where σ_{ij} are elements of Σ_{LW} .

The proximity matrix $\mathbf{D}$ measures dissimilarity between asset pairs based on their volatilities and correlations. Each element d_{ij} represents the distance between assets i and j , calculated from the variance of their return differences. Highly correlated assets with similar volatilities have smaller distances, while uncorrelated or negatively correlated assets have larger distances. This metric includes the individual variances of assets i and j (diagonal elements σ_{ii} and σ_{jj}) and their covariance (off-diagonal element σ_{ij}). Thus, assets with high positive correlation and similar risk profiles are closer together, while those with low correlation or differing risks are farther apart.

This metric satisfies three fundamental mathematical properties that ensure its validity as a distance measure:

- (1) $d_{ij} \geq 0$ (non-negativity);
- (2) $d_{ij} = 0 \iff i = j$ (identity);
- (3) $d_{ij} \leq \max(d_{ik}, d_{kj})$ (ultrametric inequality).

The non-negativity property states that distances are either positive or zero, meaning dissimilarity cannot be negative. The identity property requires that the distance between an asset and itself is zero, while distances between different assets are positive. The ultrametric inequality is a stricter form of the triangle inequality, supporting hierarchical clustering and allowing for a meaningful dendrogram.

3.3.2. Hierarchical Clustering Scheme (HCS)

Hierarchical clustering forms a tree-like structure of asset relationships by merging the most similar clusters using Johnson's (1967) algorithm for efficiency:

- (1) Initialize clusters: $\mathcal{C}_0 = \{\{1\}, \{2\}, \dots, \{n\}\}$;
- (2) At each step k , merge clusters A and B with minimal ultrametric distance:

$$d(A, B) = \max_{\substack{i \in A \\ j \in B}} d_{ij} \quad (16)$$

- (3) Update proximity matrix by removing rows/columns for A and B , adding new row/column for $A \cup B$.

The algorithm begins with each asset as its own cluster, denoted as $\mathcal{C}_0 = \{\{1\}, \{2\}, \dots, \{n\}\}$. During each iteration k , it identifies the two clusters A and B with the smallest inter-cluster distance, defined by $d(A, B) = \max_{\substack{i \in A \\ j \in B}} d_{ij}$. This complete linkage criterion promotes compact clusters by limiting the maximum distance between merged assets.

This method is effective for financial applications, as it results in stable clusters that reflect similar risk traits while preventing outliers from distorting the results. After each merge, the proximity matrix is updated by removing the merged clusters and adding a new entry for the combined cluster $A \cup B$, ensuring efficient calculations for future iterations.

The hierarchical clustering creates a sequence of nested partitions $\mathcal{C}_0 \prec \mathcal{C}_1 \prec \dots \prec \mathcal{C}_{n-1}$, which allows analysts to observe asset relationships from individual to collective groupings. This representation aids in tactical asset allocation and portfolio construction, and visualizing it with a dendrogram reveals important correlations and groupings within the asset universe.

In practice with R, I can adjust the number of clusters to ensure multiple stocks are included in each. The ideal number of clusters varies with the total stock count, whether it is 20, 30, 40, 50, or 100. I can provide detailed information on stock assignments for improved understanding of the clusters.

3.3.3. PSO with Stretching for Cluster-Aware Optimization

Cluster-Constrained Markowitz Problem

The cluster-constrained Markowitz formulation improves classical mean-variance optimization by incorporating hierarchical clustering constraints. With m clusters $G_1, \dots, G_m$, it aims to minimize portfolio variance while targeting specific expected returns and ensuring cluster-level diversification. This method mitigates concentration risk by limiting the total weight of assets in each cluster. For m clusters $G_1, \dots, G_m$ identified via HCS:

$$\min_{\mathbf{w}} \quad \frac{1}{2} \mathbf{w}^\top \Sigma_{\text{LW}} \mathbf{w} \quad (17)$$

$$\text{subject to} \quad \mathbf{w}^\top \boldsymbol{\mu} = \mu_{\text{target}} \quad (18)$$

$$\mathbf{w}^\top \mathbf{1} = 1 \quad (19)$$

$$\sum_{i \in G_k} w_i \leq \theta_k, \quad k = 1, \dots, m \quad (20)$$

The objective is to minimize portfolio variance using the Ledoit–Wolf shrinkage covariance estimator Σ_{LW} for better estimation accuracy in portfolio optimization. I have a target expected return μ_{target} , a constraint that portfolio weights sum to one, and a limit on asset weights within each cluster G_k based on a risk budget $\theta_k \in \mathbb{R}^+$ to enhance diversification.

To tackle this cluster-constrained portfolio optimization, I apply particle swarm optimization (PSO) with function stretching. This approach navigates the complex constraints of cluster diversification and mitigates issues of local optima, common in traditional optimization due to asset correlations.

Cluster-Guided Stretching

The stretching mechanism improves convergence by modifying the objective function's landscape during stagnation. This is particularly beneficial in portfolio optimization, where diverse asset clusters lead to complex, multi-modal objective functions with numerous local optima. The stretching transformation is applied according to cluster membership data to align with the asset universe's structure. When the optimization algorithm hits a local minimum at iteration τ , the objective function is recalibrated using the following formulation:

$$f_{\text{stretched}}(\mathbf{w}) = f(\mathbf{w}) + \gamma \tanh\left(\beta \|\mathbf{w} - \mathbf{w}^{(\tau)}\|\right) \cdot \mathbb{1}_{\{\mathbf{w} \in \mathcal{N}(\mathbf{w}^{(\tau)})\}} \quad (21)$$

The augmented objective function integrates the original function $f(\mathbf{w})$ with a stretching term based on the distance to a stagnation point $\mathbf{w}^{(\tau)}$. The parameter γ controls the stretching intensity, while β adjusts its sensitivity to proximity to the stagnation point. A hyperbolic tangent function provides a smooth stretching effect, preventing optimization landscape distortion.

The neighborhood $\mathcal{N}(\mathbf{w}^{(\tau)})$ is defined by cluster memberships, so the stretching is applied only to solutions within the same cluster as the stagnation point. The indicator function $\mathbb{1}_{\{\mathbf{w} \in \mathcal{N}(\mathbf{w}^{(\tau)})\}}$ restricts stretching to this area, ensuring the optimization process remains intact while exploring new allocation strategies linked to market segments.

3.3.4. Pseudo-Code for Hierarchical Cluster-Guided PSO with Stretching

In Table A4 (see the Appendix A to save space), the pseudo-code uses hierarchical clustering and particle swarm optimization to enhance portfolio construction within realistic constraints. It also overcomes traditional mean-variance limitations by resorting to the stretching function, leading to better asset diversification through cluster-based constraints and metaheuristic techniques.

3.3.5. Computational Advantages

The hierarchical clustering approach with particle swarm optimization significantly improves high-dimensional portfolio optimization by enhancing efficiency and solution quality through dimensionality reduction and regularized covariance estimation (see Table A2 in the Appendix A to save space).

The hierarchical clustering component reduces optimization dimensionality from $\mathcal{O}(n^2)$ to $\mathcal{O}(m^2)$, where m is the number of clusters. This significant reduction eases the computational burden of calculating covariance matrices in large-scale portfolio optimization and groups assets with similar risk–return profiles. Implementing Ledoit–Wolf shrinkage enhances covariance matrix estimation, addressing substantial errors typical in high dimensions and stabilizing clustering performance, particularly in volatile markets.

The particle swarm optimization (PSO) framework, with function stretching, effectively navigates local minima within the complex optimization landscape imposed by cluster

constraints. Unlike traditional gradient-based methods, PSO explores the solution space more effectively, avoiding local optima.

This hybrid approach combines robust shrinkage estimators, structured hierarchical clustering, and global PSO optimization, tackling various computational challenges to improve portfolio management efficiency and effectiveness.

3.4. K-Means Clustering of Assets as an Input to Markowitz Portfolio Optimization

3.4.1. K-Means

K-means clustering represents a fundamental unsupervised learning technique that partitions the asset universe into distinct groups based on similarity measures (Likas et al., 2003). Given K clusters with centroids $\{\mu_1, \dots, \mu_K\}$, the K-means objective seeks to minimize the total within-cluster sum of squares, effectively creating compact and well-separated clusters:

$$\underset{\{C_k\}, \{\mu_k\}}{\text{minimize}} \quad J = \sum_{k=1}^K \sum_{\mathbf{x}_i \in C_k} |\mathbf{x}_i - \mu_k|^2 \quad (22)$$

This objective function balances the dual goals of minimizing intra-cluster variance while maximizing inter-cluster separation. In this formulation, C_k represents the set of indices for points assigned to cluster k , and μ_k denotes the centroid of cluster k . The optimization simultaneously determines both the cluster assignments and the optimal centroid positions, making K-means a particularly challenging non-convex optimization problem.

3.4.2. Distance Metrics

The choice of distance metric significantly influences the clustering structure and the resulting portfolio composition. The standard Euclidean distance serves as the default distance metric used in K-means clustering:

$$d_{\text{Euclidean}}(\mathbf{x}_i, \mu_k) = |\mathbf{x}_i - \mu_k|_2 \quad (23)$$

This metric assumes that all dimensions contribute equally to the similarity measure, which may not always be appropriate for financial data, where certain risk factors or return characteristics may be more significant than others. The Euclidean distance implicitly creates spherical clusters, which can be limiting when dealing with assets that exhibit non-spherical correlation structures.

3.4.3. K-Means Clustering Scheme (KCS)

The K-means clustering scheme employs Hartigan and Wong's (1979) algorithm through an iterative refinement process that alternates between cluster assignment and centroid update steps. This algorithm provides a computationally efficient approach to solving the otherwise intractable K-means optimization problem. First, the algorithm initializes cluster centroids as $\{\mu_1^{(0)}, \mu_2^{(0)}, \dots, \mu_K^{(0)}\}$, typically through random initialization or more sophisticated methods such as K-means++. At each iteration t , the algorithm assigns each asset i to the nearest cluster according to:

$$C_k^{(t)} = \left\{ i : |\mathbf{x}_i - \mu_k^{(t)}| \leq |\mathbf{x}_i - \mu_j^{(t)}| \text{ for all } j \right\} \quad (24)$$

This assignment step ensures that each asset belongs to exactly one cluster, creating a hard partitioning of the asset universe. Subsequently, the algorithm updates centroids by computing cluster means:

$$\mu_k^{(t+1)} = \frac{1}{|C_k^{(t)}|} \sum_{i \in C_k^{(t)}} \mathbf{x}_i \quad (25)$$

The centroid update step repositions each cluster center to the geometric mean of its assigned points, minimizing the within-cluster sum of squares for the current assignment. The resulting KCS converges to a partition $\{C_1, C_2, \dots, C_K\}$ that represents a local minimum of the within-cluster sum of squares objective function.

3.4.4. K-Means Constrained Markowitz Problem

The integration of K-means clustering into the Markowitz framework creates a novel constrained optimization problem that incorporates structural information about asset relationships.

For K clusters $C_1, \dots, C_K$ identified via KCS, the constrained optimization problem becomes:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2} \mathbf{w}^\top \Sigma_{LW} \mathbf{w} \\ \text{subject to} \quad & \mathbf{w}^\top \boldsymbol{\mu} = \mu_{\text{target}} \\ & \mathbf{w}^\top \mathbf{1} = 1 \\ & \sum_{i \in C_k} w_i \leq \zeta_k, \quad k = 1, \dots, K \end{aligned} \quad (26)$$

This formulation extends the classical mean-variance optimization by introducing cluster-level constraints that prevent excessive concentration within any single asset group. Here, $\zeta_k \in \mathbb{R}^+$ represent the K-means cluster-level risk budgets, which can be determined through risk budgeting principles or regulatory requirements. These constraints ensure that the optimization process respects the underlying asset structure identified through clustering, potentially leading to more stable and interpretable portfolio allocations.

3.4.5. PSO with Stretching for K-Means Cluster-Aware Optimization

The incorporation of particle swarm optimization with stretching techniques addresses the computational challenges associated with solving the constrained K-means Markowitz problem. For each particle p representing portfolio weights $\mathbf{w}_p$, the particle swarm optimization algorithm updates velocities and positions according to:

$$\begin{aligned} \mathbf{v}_p^{(t+1)} &= \omega \mathbf{v}_p^{(t)} + c_1 r_1 (\mathbf{w}_p^{\text{local}} - \mathbf{w}_p^{(t)}) + c_2 r_2 (\mathbf{w}^{\text{global}} - \mathbf{w}_p^{(t)}) \\ \mathbf{w}_p^{(t+1)} &= \text{Proj}_{\mathcal{K}}(\mathbf{w}_p^{(t)} + \mathbf{v}_p^{(t+1)}) \end{aligned} \quad (27)$$

This swarm-based approach combines local and global search capabilities, where particles explore the feasible space while sharing information about promising regions. In this formulation, $\text{Proj}_{\mathcal{K}}$ represents the projection operator that enforces K-means cluster constraints and budget constraints, ensuring that all generated solutions remain feasible throughout the optimization process.

3.4.6. K-Means Cluster-Guided Stretching

The stretching technique provides a sophisticated mechanism for escaping local minima by temporarily modifying the objective function landscape. When local minima occur at iteration τ , the algorithm applies a stretching technique defined by:

$$f_{\text{stretched}}^{\text{KM}}(\mathbf{w}) = f(\mathbf{w}) + \alpha \tanh\left(\lambda |\mathbf{w} - \mathbf{w}^{(\tau)}|\right) \cdot \mathbb{1}_{\{\mathbf{w} \in \mathcal{M}(\mathbf{w}^{(\tau)})\}} \quad (28)$$

This formulation adds a repulsive force around the current local minimum, encouraging the search process to explore alternative regions of the solution space. Here, $\mathcal{M}(\mathbf{w}^{(\tau)})$ is defined via K-means cluster memberships, ensuring that the stretching effect respects the underlying asset structure. The parameters $\alpha, \lambda \in \mathbb{R}^+$ control the magnitude and range of the stretching effect, while $\mathbb{1}$ is the indicator function that activates the stretching only within the relevant cluster neighborhood.

3.4.7. Portfolio Insights

K-means clustering enhances portfolio analysis by identifying asset groupings that reveal correlations often masked by individual asset weights. It allows for better segmentation based on risk–return profiles, helping portfolio managers evaluate the contributions of different asset classes to overall risk and return.

This method also highlights concentration risks from over-weighted similar assets, informing diversification strategies for a more balanced allocation. Additionally, it supports effective risk budgeting and portfolio rebalancing by pinpointing reallocations needed as asset characteristics change. Clustering further improves factor exposure analysis by grouping assets with similar sensitivities, ensuring thorough exposure to systemic risk.

3.4.8. Pseudo-Code for K-Means Cluster-Guided PSO with Stretching

In Table A5 (see the Appendix A to save space), the pseudo-code uses K-means clustering to group stocks into homogeneous clusters based on return patterns, reducing the dimensionality of the portfolio optimization problem. The cluster weights are then determined using an enhanced particle swarm optimization algorithm with adaptive stretching transformations and time-varying parameters to maximize the Sharpe ratio, before distributing the optimal weights proportionally to individual stocks within each cluster.

3.5. DBSCAN Clustering of Assets as an Input to Markowitz Portfolio Optimization

The DBSCAN algorithm requires two main parameters for effective clustering (Khan et al., 2014). The first parameter is ε (Eps), which represents the neighborhood radius that defines the maximum distance between two points for them to be considered neighbors. The second parameter is MinPts, which specifies the minimum number of points required to form a dense region or cluster.

Parameter selection can be guided by portfolio characteristics, the desired granularity of clusters, and domain expertise in financial markets. These considerations help determine appropriate values that align with the specific characteristics of the financial data being analyzed.

3.5.1. Core Concepts

ε -Neighborhood

The ε -neighborhood of point $\mathbf{x}_i$ is defined as the set of all points within the specified radius distance. This concept is mathematically expressed as:

$$N_\varepsilon(\mathbf{x}_i) = \{\mathbf{x}_j \in X \mid d(\mathbf{x}_i, \mathbf{x}_j) \leq \varepsilon\} \quad (29)$$

This equation captures all points $\mathbf{x}_j$ in the dataset X that are within distance ε from the reference point $\mathbf{x}_i$.

Core Point

A point $\mathbf{x}_i$ is classified as a core point when it satisfies specific density requirements. The mathematical condition for a core point is:

$$|N_\varepsilon(\mathbf{x}_i)| \geq \text{MinPts} \quad (30)$$

This means that a point becomes a core point if the number of points in its ε -neighborhood is at least MinPts, indicating sufficient local density to anchor a cluster.

Directly Density-Reachable

A point $\mathbf{x}_j$ is directly density-reachable from $\mathbf{x}_i$ when it satisfies two simultaneous conditions. The mathematical formulation is:

$$\mathbf{x}_j \in N_\varepsilon(\mathbf{x}_i) \quad \text{and} \quad \mathbf{x}_i \text{ is a core point} \quad (31)$$

This relationship establishes a direct connection between points based on proximity and the core point status of the reference point.

Density-Reachable

A point $\mathbf{x}_j$ is density-reachable from $\mathbf{x}_i$ when there exists a chain of points connecting them through direct density-reachability. The formal definition requires a sequence $(\mathbf{x}_i = \mathbf{x}_{p_1}, \mathbf{x}_{p_2}, \dots, \mathbf{x}_{p_n} = \mathbf{x}_j)$ such that:

$$\mathbf{x}_{p_{k+1}} \text{ is directly density-reachable from } \mathbf{x}_{p_k} \text{ for } k \in 1, 2, \dots, n-1 \quad (32)$$

This transitive relationship allows points to be connected through intermediate core points, extending the reach of cluster formation beyond immediate neighborhoods.

Density-Connected

Two points $\mathbf{x}_i$ and $\mathbf{x}_j$ are density-connected when they can both reach a common point through density-reachability. The mathematical condition states:

$$\text{Both } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are density-reachable from } \mathbf{x}_k \quad (33)$$

This symmetric relationship forms the foundation for grouping points into the same cluster, even when they may not be directly reachable from each other.

DBSCAN Algorithm

Given parameters ε (neighborhood radius) and MinPts (minimum points), the DBSCAN objective focuses on identifying density-based clusters by grouping points that are closely packed together. The optimization problem can be formulated as:

$$\underset{C_1, \dots, C_K}{\text{maximize}} \quad \sum_{k=1}^K \sum_{\mathbf{x}_i \in C_k} |N_\varepsilon(\mathbf{x}_i) \cap C_k| \quad (34)$$

In this formulation, C_k represents the set of indices for points assigned to cluster k , and $N_\varepsilon(\mathbf{x}_i) = \{\mathbf{x}_j \in X \mid d(\mathbf{x}_i, \mathbf{x}_j) \leq \varepsilon\}$ defines the ε -neighborhood of point $\mathbf{x}_i$.

Distance Metrics

The default distance metric used in DBSCAN is the standard Euclidean distance. This metric is mathematically expressed as:

$$d_{\text{Euclidean}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 \quad (35)$$

This L_2 norm provides a natural measure of similarity between points in the feature space.

3.5.2. DBSCAN Clustering Scheme (DCS)

The DBSCAN clustering scheme follows the algorithm developed by Ester et al. (1996) and proceeds through several systematic steps. Initially, for each point $\mathbf{x}_i$, the algorithm determines whether it qualifies as a core point using the criterion:

$$\text{Core}(\mathbf{x}_i) = \begin{cases} \text{True} & \text{if } |N_\epsilon(\mathbf{x}_i)| \geq \text{MinPts} \\ \text{False} & \text{otherwise} \end{cases} \quad (36)$$

Subsequently, for each core point $\mathbf{x}_i$, the algorithm creates a cluster by identifying all density-reachable points:

$$C_k = \{\mathbf{x}_j \in X : \mathbf{x}_j \text{ is density-reachable from } \mathbf{x}_i\} \quad (37)$$

Finally, the algorithm classifies remaining points as noise if they are not density-reachable from any core point. The resulting DCS produces clusters $C_1, C_2, \dots, C_K$ and a noise set $\mathcal{N}$ containing outliers.

3.5.3. DBSCAN Constrained Markowitz Problem

For K clusters $C_1, \dots, C_K$ identified via DCS, the optimization problem incorporates cluster-aware constraints. The formulation is:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2} \mathbf{w}^\top \Sigma L \mathbf{W} \mathbf{w} \\ \text{subject to} \quad & \mathbf{w}^\top \boldsymbol{\mu} = \mu_{\text{target}} \\ & \mathbf{w}^\top \mathbf{1} = 1 \\ & \sum_{i \in C_k} w_i \leq \zeta_k, \quad k = 1, \dots, K \\ & \sum_{i \in \mathcal{N}} w_i \leq \eta \end{aligned} \quad (38)$$

In this constrained optimization problem, $\zeta_k \in \mathbb{R}^+$ represents DBSCAN cluster-level risk budgets, and $\eta \in \mathbb{R}^+$ denotes the noise risk budget.

3.5.4. PSO with Stretching for DBSCAN Cluster-Aware Optimization

For each particle p representing portfolio weights $\mathbf{w}_p$, the particle swarm optimization dynamics are governed by velocity and position updates. The velocity update equation is:

$$\begin{aligned} \mathbf{v}_p^{(t+1)} &= \omega \mathbf{v}_p^{(t)} + c_1 r_1 (\mathbf{w}_p^{\text{local}} - \mathbf{w}_p^{(t)}) + c_2 r_2 (\mathbf{w}^{\text{global}} - \mathbf{w}_p^{(t)}) \\ \mathbf{w}_p^{(t+1)} &= \text{ProjD}(\mathbf{w}_p^{(t)} + \mathbf{v}_p^{(t+1)}) \end{aligned} \quad (39)$$

The projection operator ProjD enforces DBSCAN cluster constraints, noise constraints, and budget constraints simultaneously.

3.5.5. DBSCAN Cluster-Guided Stretching

When local minima occur at iteration τ , the stretching function modifies the objective landscape. The stretched objective function is defined as:

$$f_{\text{stretched}}^{\text{DB}}(\mathbf{w}) = f(\mathbf{w}) + \phi \tanh(\psi |\mathbf{w} - \mathbf{w}^{(\tau)}|) \cdot \mathbb{1}_{\mathbf{w} \in \mathcal{R}(\mathbf{w}^{(\tau)})} \quad (40)$$

In this formulation, $\mathcal{R}(\mathbf{w}^{(\tau)})$ is defined via DBSCAN cluster memberships and noise classification, $\phi, \psi \in \mathbb{R}^+$ are stretching parameters, and $\mathbb{1}$ represents the indicator function that activates the stretching effect only within the specified region.

3.5.6. Pseudo-Code for DBSCAN Cluster-Guided PSO with Stretching

In Table A6 (see the Appendix A to save space), the pseudo-code merges density-based clustering for stock grouping with dynamic PSO parameters to locate the global optima, and assigns weights to securities for optimized portfolio construction.

4. Empirical Application

4.1. Data Preparation

This analysis examines a six-year period from January 2020 to December 2025, utilizing daily data. For each stock market under consideration, I collect for the purpose of the empirical analysis: (i) the Stock Index itself; (ii) the Constituents of the Index; (iii) 10-year government bond yields as the risk-free rate; and (iv) one Exchange Traded Fund (ETF) per stock market. The objective is to create optimized buy-and-hold Markowitz portfolios over the six-year period and evaluate the performance of PSO with stretching and clustering against market indices and ETF benchmarks.

To ensure numerical stability in high-dimensional settings, I implement a safe covariance shrinkage function. This mechanism checks the observation-to-asset ratio and applies James-Stein type shrinkage to the covariance matrix; in cases where data density is insufficient, the algorithm automatically falls back to a robust standard covariance estimator to prevent singular matrix errors during PSO fitness evaluation.

4.2. DJ Euro Stoxx 50

The Euro Stoxx 50 is a benchmark for the eurozone equity market, comprising the 50 largest companies across various industries.

In Table 1, the portfolio optimization analysis of the Eurostoxx 50 reveals significant performance differences among algorithms. K-means clustering is the most effective, achieving a Sharpe ratio of 0.7913, exceeding the SWARM algorithm at 0.7865, while Hierarchical Clustering and DBSCAN lag behind. K-means provides the highest portfolio return mean of 0.1008, despite having a moderate risk standard deviation of 0.1814. In terms of downside risk, K-means has a Calmar ratio of 0.3057, outperforming other methods and demonstrating strong tail risk protection along with substantial returns. It reports a maximum drawdown of 0.2721, balancing risk and return effectively. Additional metrics further support K-means, with a Value-at-Risk of -0.2810 , Expected Shortfall of -0.3544 , and the best Omega ratio at 1.4763. Overall, K-means clustering is the top choice for portfolio optimization in the Eurostoxx 50, excelling in both returns and risk management.

Table 1. Portfolio Analytics for DJ Euro Stoxx 50.

Method	SWARM	Hierarchical Clustering	K-Means	DBSCAN
Portfolio_Return_Mean	0.0856	0.0462	0.1008	−0.9650
Portfolio_Risk_SD	0.1633	0.1756	0.1814	4.2108
Sharpe_Ratio	0.7865	0.5067	0.7913	−0.2190
Omega_Ratio	1.4288	1.2013	1.4763	0.4813
Beta	0.0520	0.0700	0.8401	0.6179
Treynor_Ratio	3.3079	2.3039	0.2096	−1.1263
Sortino_Ratio	12.2961	11.2375	11.0633	−0.6650
Jensen_Alpha	0.0497	0.0466	0.0107	−0.2304
Portfolio_Drawdown	0.2432	0.2955	0.2721	2.3465
Calmar_Ratio	0.2940	0.1032	0.3057	−0.1213
VaR_95	−0.2573	−0.2748	−0.2810	−3.4641
ES_95	−0.3375	−0.3414	−0.3544	−3.4641
Skewness	−0.3351	−0.0118	−0.2368	−1.0034
Kurtosis	0.0939	−0.2214	−0.2666	4.7361

In Table A7 (see Appendix A to save space), the portfolio created using cluster-guided PSO with K-Means features a two-tier weighting structure. The top 15 holdings are split into two clusters: Cluster 5 and Cluster 7, each with different risk allocations. Cluster 5, the highest-tier, contains six assets—Adyen, Hermès, Ferrari, Schneider Electric, ASML Holding, and Dassault Systèmes—each weighted equally at 6.73%, totaling 40.39% of the portfolio. This uniform weighting optimizes the assets for better risk–return potential. Cluster 7 includes nine assets, with individual allocations of 4.49%, contributing to 40.38% of the portfolio. Notable companies here are Enel, Iberdrola, and SAP, providing sector diversification, albeit with potentially lower risk-adjusted returns than Cluster 5. From a diversification standpoint, the portfolio spans various sectors, but it carries concentration risks due to uniform weights within the clusters, which may lead to correlated performance in market downturns. The constraints of the algorithm prioritize uniform weights, illustrating a trade-off between structured risk management and traditional mean-variance optimization, thereby impacting the diversification efficiency.

As illustrated in Figure 1, the K-Means guided PSO with stretching slightly outperforms the performance of both the DJ Eurostoxx 50 market index itself, as well as the Amundi EUR STOXX 50 Dly (-2x) Inv UCITS ETF Acc (BXX).

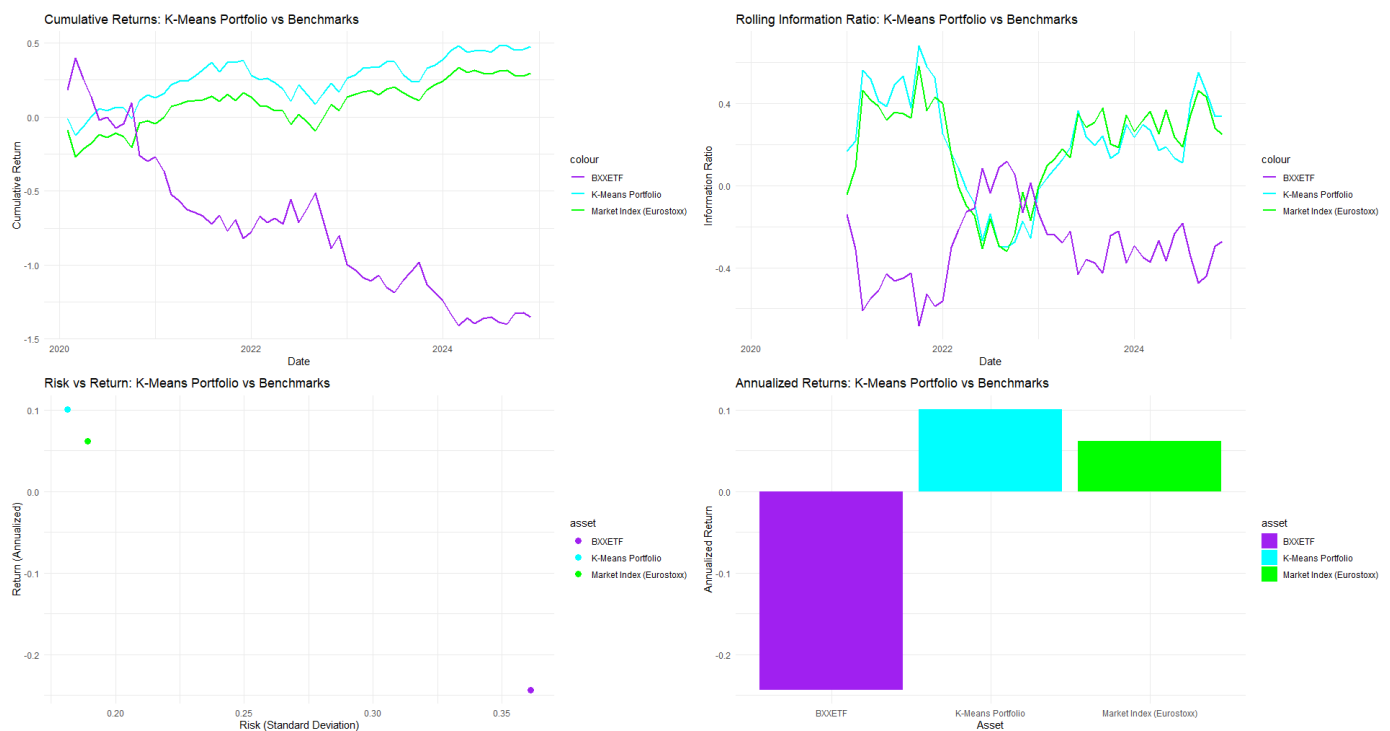


Figure 1. Performance analytics for DJ Euro Stoxx 50.

4.3. S&P CNX Nifty 50

Exchanged on India’s National Stock Exchange (NSE), the Nifty 50 is an important index made up of 50 selected stocks from major economic sectors.

In Table 2, both the SWARM algorithm (without clustering) and the Hierarchical Clustering algorithm perform well in terms of portfolio optimization techniques for the Nifty 50 index. Both methods yield similar Sharpe ratios, with SWARM at 1.06571 and Hierarchical Clustering at 1.06653. However, SWARM indicates a higher mean portfolio return of 0.22292, surpassing Hierarchical Clustering’s 0.19703; though it incurs more risk, evident in the standard deviations of 0.20661 for SWARM versus 0.18218 for Hierarchical Clustering. In terms of downside risk management, Hierarchical Clustering excels with a Calmar ratio of 0.74068 compared to 0.64965 for SWARM, alongside a maximum drawdown

of 0.24009 versus 0.29790. Value-at-Risk statistics further favor Hierarchical Clustering, showing VaR_{95} at -0.25785 compared to SWARM's -0.34271 , and better Expected Shortfall at -0.42925 against SWARM's -0.59912 . Additionally, Hierarchical Clustering's Sortino ratio of 1.29239 outperforms SWARM's 1.14218, indicating superior risk-adjusted returns. It also demonstrates lower negative skewness (-0.41923 versus -1.57752) and decreased kurtosis (2.16304 versus 4.34262), suggesting a more stable return distribution. In comparison, both K-means (0.76777) and DBSCAN (1.18035) present competitive Sharpe ratios but struggle with downside risk management, evidenced by lower Calmar ratios of 0.38237 and -0.02367 . K-means has the highest maximum drawdown at 0.37850 and concerning negative skewness at -2.21309 . In conclusion, Hierarchical Clustering is recommended for optimizing Nifty 50 portfolios due to its superior Calmar ratio and effective downside risk management, essential for practical investment strategies.

In Table A8 (see Appendix A to save space), the Nifty 50 portfolio, organized through Hierarchical Clustering, emphasizes the top eight holdings, each at 7.17%, leading to a total of 57.38% from Cluster 4. Sector composition shows a mix of benefits and risks. The pharmaceutical sector, led by Cipla Ltd. and Dr. Reddy's Laboratories Ltd., accounts for 14.35%, enhancing diversity but facing regulatory challenges. The technology sector, featuring HCL Technologies Ltd. and Wipro Ltd., also at 14.35%, is exposed to global tech spending shifts and currency risks. In the industrial and materials sectors, Hindalco Industries Ltd. and Bharat Petroleum Corporation Ltd. contribute 14.35%, linking them to commodity prices. Tata Motors Ltd. in automotive and Britannia Industries Ltd. in consumer goods dilute risk, each at 7.17%. The portfolio has a balanced sector distribution but suffers from equal weighting, which contradicts the Markowitz principle of tailored allocations based on expected returns and risks. The focus on Cluster 4 increases risk during market stress, and while sector diversity helps, uniform weighting can limit risk-adjusted returns.

Table 2. Portfolio Analytics for Nifty 50.

Method	SWARM	Hierarchical Clustering	K-Means	DBSCAN
Portfolio_Return_Mean	0.22292	0.19703	0.17701	4.91076
Portfolio_Risk_SD	0.20661	0.18218	0.22700	4.15811
Sharpe_Ratio	1.06571	1.06653	0.76777	1.18035
Omega_Ratio	2.12657	2.16457	1.83345	1.63854
Beta	-0.03970	-0.06827	1.07518	0.52324
Treynor_Ratio	-1.23745	-0.62771	0.03527	1.03848
Sortino_Ratio	1.14218	1.29239	0.75353	0.77486
Jensen_Alpha	0.01418	0.01237	0.00146	0.15224
Portfolio_Drawdown	0.29790	0.24009	0.37850	1.22714
Calmar_Ratio	0.65965	0.74068	0.38237	-0.02367
VaR_95	-0.34271	-0.25785	-0.40076	-3.46410
ES_95	-0.59912	-0.42925	-0.90116	-3.46410
Skewness	-1.57752	-0.41923	-2.21309	0.27688
Kurtosis	4.34262	2.16304	9.45818	6.58319

As pictured in Figure 2, both the Hierarchical Clustering-guided PSO with stretching algorithm and the Nippon Nifty 50 BeES (NBES) ETF beat the market index; the clustering algorithm even surpasses the performance of the ETF itself.



Figure 2. Performance analytics for Nifty 50.

4.4. FTSE China A50

The China A50 Index tracks 50 major A-shares from the Shanghai and Shenzhen Stock Exchanges, developed by FTSE Russell. It serves as a benchmark for equity investments in mainland China and reflects increasing global interest in Chinese equities.

In Table 3, DBSCAN clustering demonstrates strong performance for the China A50 index, making it advantageous for portfolio optimization. It achieved a Sharpe ratio of 0.4077, similar to K-means' 0.4100, while significantly outperforming SWARM (0.0631) and Hierarchical Clustering (0.2789). DBSCAN also had the highest Calmar ratio at 0.5790, exceeding K-means (0.5202), SWARM (0.1417), and Hierarchical Clustering (0.2095), indicating better risk-adjusted returns relative to maximum drawdown. The mean return for DBSCAN reached 2.5397, far surpassing K-means (0.1556) and SWARM (0.0372), highlighting its ability to identify high-return opportunities in the Chinese market. Additionally, it recorded the highest Treynor ratio (0.0859) and a favorable Sortino ratio (0.2526), in stark contrast to the negative ratios from competing methods, indicating better downside risk management. DBSCAN also produced the highest Jensen's alpha (0.1692), showing significant value beyond market expectations. While its volatility metrics are higher, this reflects a proactive return-seeking strategy that captures the dynamic nature of Chinese A-shares. The density-based approach of DBSCAN's superior Calmar ratio and competitive Sharpe ratio make it a preferred choice for investors aiming to maximize risk-adjusted returns in the Chinese equity market.

In Table A9 (see the Appendix A to save space), the analysis of DBSCAN-Guided PSO with the Stretching Algorithm for China A50 stocks reveals a significant deviation from the Markowitz portfolio approach, indicating strong sector concentration. The top nine holdings are primarily in the consumer discretionary sector, especially the alcoholic beverage sub-sector, comprising about 39.7% of the portfolio. Key stocks include Kweichow Moutai and Wuliangye Yibin, signifying strong co-movement in the baijiu industry. There is minimal representation from healthcare and technology, raising concerns about diversification. The algorithm allocates nearly 9.92% to the top holdings, focusing on risk-adjusted returns instead of traditional diversification, suggesting a similar risk–return

profile among these assets. This concentration in consumer staples and discretionary sectors contrasts with the Markowitz principle of including uncorrelated assets to mitigate risk. The findings imply that correlation-based strategies may be less effective in the Chinese market due to its unique sector dynamics.

Table 3. Portfolio Analytics for China A50.

Method	SWARM	Hierarchical Clustering	K-Means	DBSCAN
Portfolio_Return_Mean	0.0372	0.1380	0.1556	2.5397
Portfolio_Risk_SD	0.2192	0.4111	0.3226	6.1724
Sharpe_Ratio	0.0631	0.2789	0.4100	0.4077
Omega_Ratio	1.1418	1.3168	1.4002	1.2243
Beta	0.2081	0.1498	1.0405	3.5352
Treynor_Ratio	−0.3386	−0.2903	−0.0375	0.0859
Sortino_Ratio	−1.2144	−0.4245	−0.5082	0.2526
Jensen_Alpha	−0.0203	−0.0126	0.0127	0.1692
Portfolio_Drawdown	0.2627	0.6591	0.2992	4.3862
Calmar_Ratio	0.1417	0.2095	0.5202	0.5790
VaR_95	−0.3702	−0.7638	−0.5360	−3.4641
ES_95	−0.5768	−1.1845	−0.7963	−3.4641
Skewness	−0.5380	−1.5406	−0.7060	−0.1624
Kurtosis	2.0075	3.5843	1.4290	4.5261

As shown in Figure 3, the DBSCAN method effectively identifies high-quality Chinese equities and may exploit market inefficiencies better than traditional optimization methods. It surpasses both the China A50 market index and the iShares China Large-Cap FXI ETF.

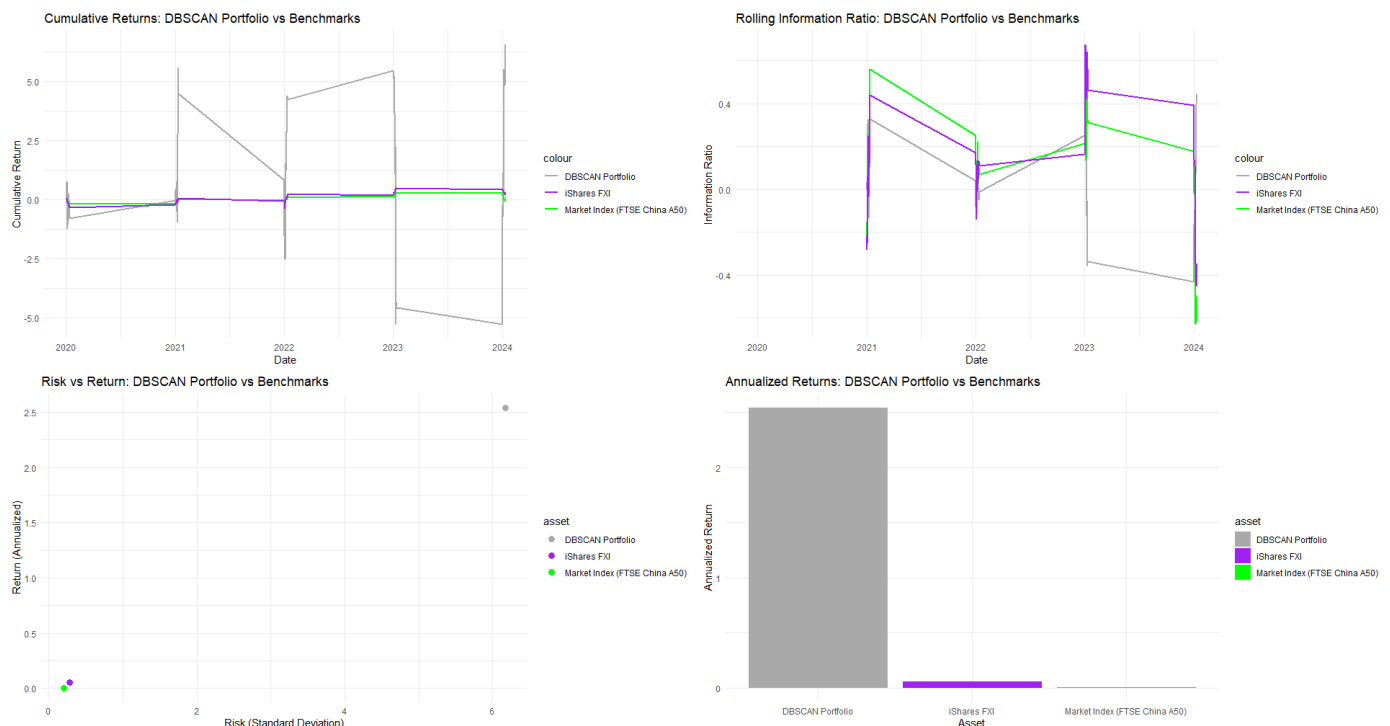


Figure 3. Performance analytics for China A50.

4.5. Euronext CAC 40

The CAC 40 is the primary index for the Paris Bourse, featuring the performance of the 40 major companies in the French equity market.

In Table 4, regarding the French CAC 40 stock market, the SWARM Particle Swarm Optimization (PSO) with the Stretching algorithm significantly outperforms clustering-guided PSO methods across key risk-adjusted performance metrics. Notably, the SWARM algorithm (without clustering) records a Sharpe ratio of 0.4690. The Calmar ratio of 0.5406 indicates strong performance against maximum drawdown. The unconstrained SWARM PSO with Stretching proves more effective for portfolio optimization, balancing risk with a beta of 1.2077 and a maximum drawdown of 0.2594. Its favorable Sharpe and Calmar ratios demonstrate a solid return-risk balance, and return distributions show positive skewness. The success of the SWARM PSO with Stretching supports the use of advanced algorithms that do not necessarily rely on predefined asset groupings. In conclusion, although clustering strategies may appear advantageous, they can impose constraints that hinder portfolio efficiency in the French CAC 40 stock market.

In Table A10 (see the Appendix A to save space), the SWARM PSO portfolio, without clustering, reflects Markowitz diversification theory, focusing on sector distribution and risk management. The algorithm allocates 5.0403% to the top 19 holdings, indicating a balanced approach with similar risk-adjusted returns, differing from market capitalization-weighted strategies. The portfolio displays solid sector diversification in the French economy, featuring TotalEnergies (energy), Vinci (industrials), and automotive firms like Stellantis and Renault. A significant 20.16% is allocated to financials, including BNP Paribas and AXA, highlighting sector importance but raising concentration risk concerns. Major firms like Schneider Electric and Airbus provide exposure to both domestic and international markets, aligning with France's industrial focus. The portfolio includes Sanofi (pharmaceuticals), EssilorLuxottica (consumer discretionary), and Saint-Gobain (construction). While sector distribution appears reasonable, the asset allocation from the SWARM PSO algorithm somewhat lacks exposure to traditional consumer staples and technology, possibly reflecting preferences for value sectors within the CAC 40.

Table 4. Portfolio Analytics for CAC 40.

Method	SWARM	Hierarchical Clustering	K-Means	DBSCAN
Portfolio_Return_Mean	0.1651	0.1462	0.1023	0.1245
Portfolio_Risk_SD	0.2145	0.2393	0.2692	0.1738
Sharpe_Ratio	0.4690	0.3415	0.1407	0.3454
Omega_Ratio	1.7104	1.5652	1.3357	1.6255
Beta	1.2077	1.2719	1.3657	0.8837
Treynor_Ratio	−0.1481	−0.1444	−0.1428	−0.2141
Sortino_Ratio	−2.3645	−2.3082	−2.2374	−2.6110
Jensen_Alpha	0.0136	0.0157	0.0175	−0.0069
Portfolio_Drawdown	0.2594	0.2379	0.3056	0.2138
Calmar_Ratio	0.5406	0.4890	0.2150	0.5068
VaR_95	−0.2490	−0.2487	−0.3091	−0.2350
ES_95	−0.2999	−0.2931	−0.3292	−0.2853
Skewness	0.7377	1.1457	0.9884	0.2996
Kurtosis	2.1604	3.8584	3.9625	−0.2905

As displayed in Figure 4, the SWARM (without clustering) with the Stretching algorithm outperforms the CAC 40 index, and matches the performance of the Amundi CAC 40 UCITS and Lyxor UCITS Daily ETFs.

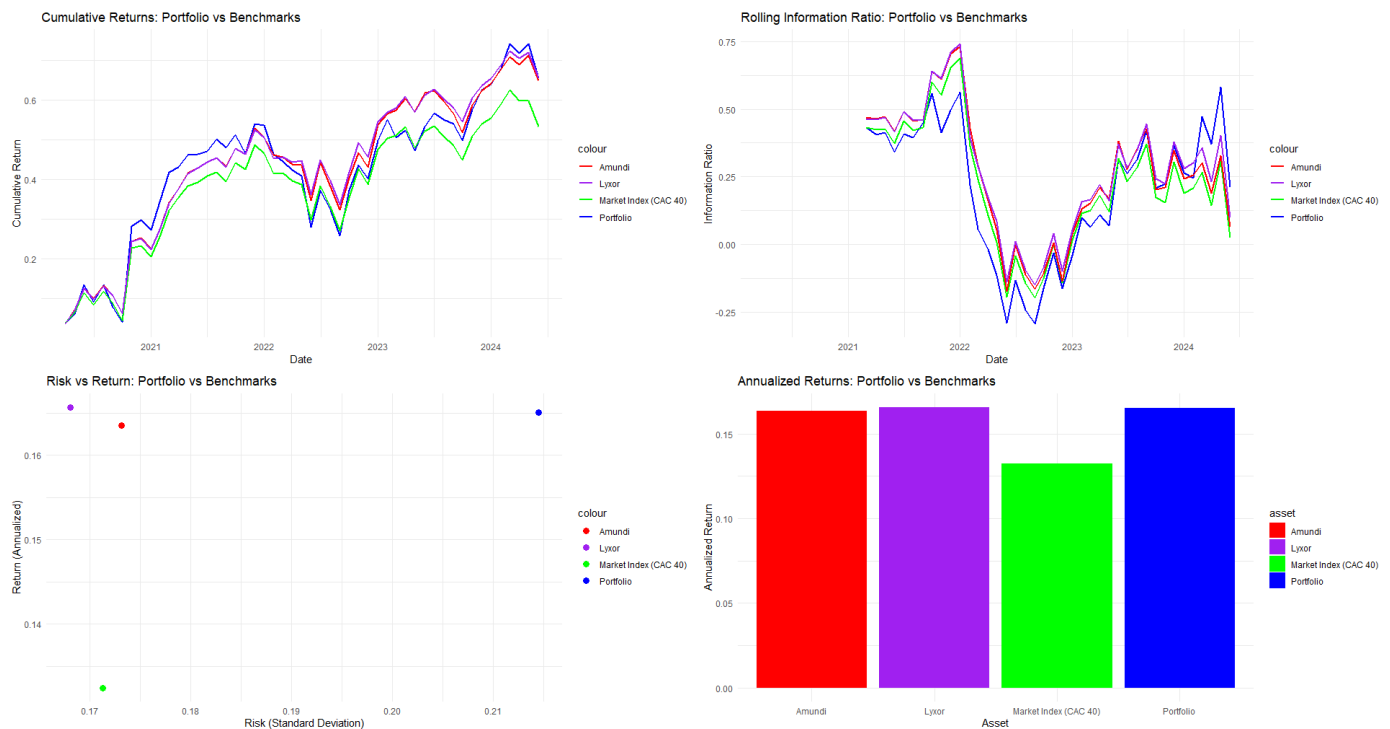


Figure 4. Performance analytics for CAC40.

4.6. Euronext BEL 20

The BEL 20 Index is the main benchmark for the Brussels Stock Exchange, representing the 20 largest and most liquid companies listed on the Belgian stock exchange, making it a key indicator of the Belgian equity market.

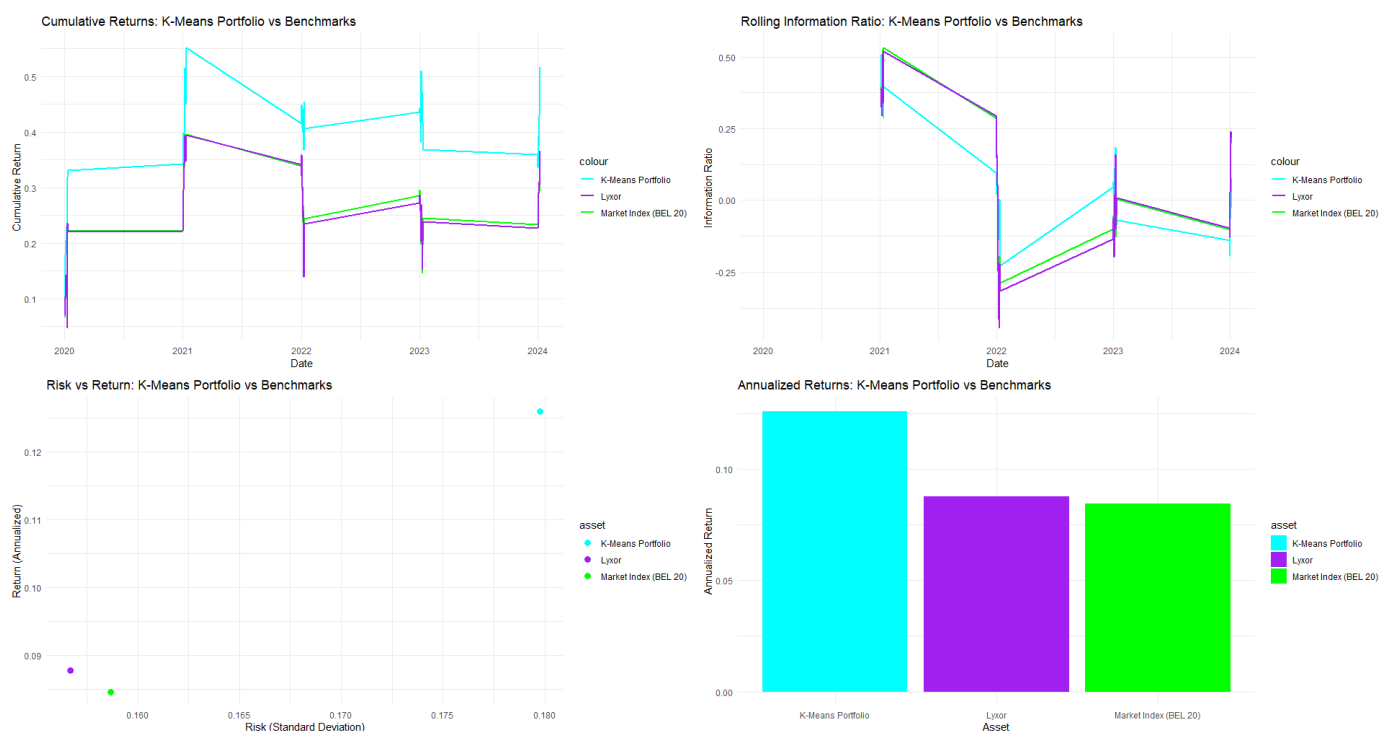
In Table 5, for the Belgian BEL 20 stock market, the K-Means-guided Particle Swarm Optimization (PSO) with the Stretching algorithm outperforms other optimization methods. It achieves a Sharpe ratio of 1.0004, exceeding the SWARM method at 0.9577 and hierarchical clustering at 0.8264. K-Means effectively balances returns with risk, with a portfolio volatility higher than SWARM's 0.1310 and hierarchical clustering's 0.1393. It offers a Calmar ratio of 0.5783, demonstrating superior performance. Despite a maximum drawdown of 0.2178, it remains manageable due to its favorable risk–return profile. The K-Means methodology has a beta of 0.8459, indicating moderate sensitivity to market movements, which allows it to capitalize on market uptrends while maintaining some defensive qualities. The K-Means algorithm shows acceptable tail risk, with a 5% Value-at-Risk of -0.2421 and an Expected Shortfall of -0.3140 , justifying its higher tail risk through significant returns.

In Table A11 (see the Appendix A to save space), regarding the BEL 20 Cluster-Based Allocation, the portfolio exhibits a high concentration risk, with 26.83% invested in arGEN-X SE, a biopharmaceutical company, which undermines diversification. Other significant asset allocations are visible at 4.48% each in Cluster 1: the portfolio is diversified across industries, including sustainability (Umicore), financial services (KBC Group, Ageas), and consumer goods (Anheuser-Busch InBev). Melexis NV adds semiconductor exposure. Other holdings include Proximus (telecommunications), Solvay (chemicals), and Aperam (specialty materials), enhancing industrial diversification. While sectors are somewhat diversified, the significant investment in arGEN-X SE does introduce considerable idiosyncratic risk. To minimize exposure to high-risk assets, the K-Means PSO utilizing the Stretching method should decrease the allocation to arGEN-X and instead reallocate based on asset characteristics that could enhance performance while mitigating risks.

Table 5. Portfolio Analytics for BEL 20.

Method	SWARM	Hierarchical Clustering	K-Means	DBSCAN
Portfolio_Return_Mean	0.0716	0.0612	0.1260	−0.3851
Portfolio_Risk_SD	0.1310	0.1393	0.1798	4.7951
Sharpe_Ratio	0.9577	0.8264	1.0004	−0.0691
Omega_Ratio	1.5105	1.4036	1.6683	0.9158
Beta	0.7123	0.7682	0.8459	12.3068
Treynor_Ratio	0.2901	0.2653	0.2613	0.0040
Sortino_Ratio	40.4204	63.1906	19.0561	0.0532
Jensen_Alpha	0.0164	0.0122	0.0125	−0.7323
Portfolio_Drawdown	0.1785	0.1749	0.2178	2.3557
Calmar_Ratio	0.4010	0.3500	0.5783	−0.1635
VaR_95	−0.1817	−0.1712	−0.2421	−3.4641
ES_95	−0.2379	−0.2095	−0.3140	−3.4641
Skewness	0.2464	0.8357	0.2675	0.6006
Kurtosis	0.9417	1.3805	0.7121	3.4294

As shown in Figure 5, the K-Means-guided PSO with Stretching improves mean-variance optimization, enabling better risk–return trade-offs compared to SWARM and other methods in the Belgian market.

**Figure 5.** Performance analytics for BEL 20.

4.7. Nasdaq OMX Copenhagen 20

The OMX Copenhagen 20 Index is the leading benchmark for the Copenhagen Stock Exchange, comprising the 20 most actively traded stocks, which tracks the performance in the Danish equity market.

In Table 6, the SWARM (without clustering) PSO with the Stretching algorithm shows superior performance in risk-adjusted return metrics for the Nasdaq OMX Copenhagen 20. It achieves a Sharpe ratio of 0.9581, outperforming clustering-based methods and demonstrating superior risk–return efficiency. With a Calmar ratio of 0.8096, SWARM balances returns and downside risk effectively, aligning with Markowitz’s goal of

maximizing return per unit of risk. The methodology yields a mean return of 0.2799 and a standard deviation of 0.2381, placing it on the efficient frontier and showcasing optimal mean-variance trade-offs. The low drawdown of 0.3457 highlights strong risk management. Additionally, the Omega ratio of 2.2149 indicates more than double the potential upside compared to downside risk. The portfolio's beta of 1.0602 suggests slightly above-market systemic risk, while the return distribution's skewness of -0.0608 and kurtosis of -0.0364 is close to normal, supporting mean-variance optimization. In conclusion, the hybrid SWARM-PSO-Stretching methodology effectively tackles Markowitz portfolio optimization challenges and represents a significant advancement in modern investment strategy for the Danish stock market.

In Table A12 (see the Appendix A to save space), the SWARM particle swarm optimization enhanced with a stretching algorithm achieves a uniform allocation of 17.86% across the top five constituents of the Nasdaq OMX Copenhagen 20 portfolio, contrasting with traditional Markowitz optimization, which often results in uneven allocations. The portfolio is well-diversified within the Danish economy. NKT A/S offers low-, medium-, and high-voltage power solutions, aligning with Europe's renewable energy transition, while Vestas Wind Systems focuses on wind turbine design. Jyske Bank, Denmark's third-largest bank, adds financial stability with regional diversification, and Pandora A/S, a prominent jewelry manufacturer, enhances geographic reach. The ROCKWOOL Group contributes through stone wool solutions, benefiting from strong construction activity. The uniform weight distribution suggests the SWARM algorithm optimizes beyond mean-variance efficiency, resulting in a balanced allocation across infrastructure, renewable energy, financial services, consumer goods, and industrial materials, ensuring strong regional coherence in the Nordic market.

Table 6. Portfolio Analytics for OMX C20.

Method	SWARM	Hierarchical Clustering	K-Means	DBSCAN
Portfolio_Return_Mean	0.2799	0.2818	0.1832	−0.8877
Portfolio_Risk_SD	0.2381	0.3742	0.3717	7.6405
Sharpe_Ratio	0.9581	0.6149	0.3537	−0.1230
Omega_Ratio	2.2149	1.6806	1.4089	0.7266
Beta	1.0602	1.0692	0.9260	−0.1473
Treynor_Ratio	−0.1012	−0.1000	−0.1408	5.1346
Sortino_Ratio	−1.5967	−1.1550	−1.2764	−0.3838
Jensen_Alpha	0.0033	0.0037	−0.0077	−0.2231
Portfolio_Drawdown	0.3457	0.5710	0.6506	1.1049
Calmar_Ratio	0.8096	0.4936	0.2817	−0.8034
VaR_95	−0.3200	−0.5247	−0.6194	−3.4641
ES_95	−0.4208	−0.7274	−0.8176	−3.4641
Skewness	−0.0608	0.0447	−0.6593	−3.6775
Kurtosis	−0.0364	1.0118	0.3968	21.8661

As reported in Figure 6, the SWARM algorithm beats the OMX Copenhagen market index itself, as well as the OMX Copenhagen Benchmark (OMXCBI) ETF by a small margin.

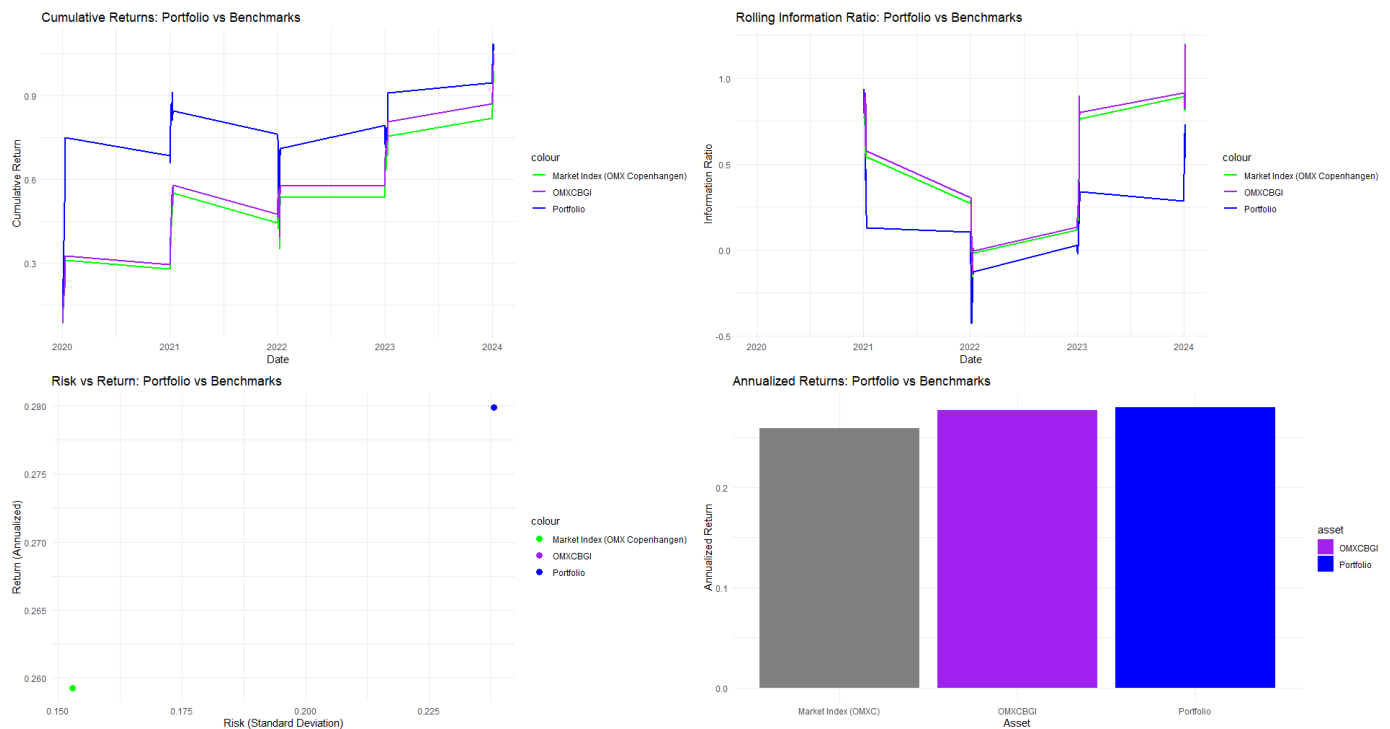


Figure 6. Performance analytics for OMX C20.

4.8. Sensitivity Analyses

In this section, I verify the robustness of our empirical estimates with respect to three sources of variations (for brevity, applied only to the Belgium and Danish stock markets with the SWARM algorithm).

4.8.1. Rebalancing

The backtesting framework employs an expanding window approach. While the initial portfolio is held for 24 months, I implement dynamic monthly rebalancing from month 25 through month 60. During each rebalancing event, the Swarm solver recalculates optimal weights based on all preceding data. To align the model with real-world trading conditions, implicit transaction costs of 3 basis points are applied to all weight rotations¹, ensuring that the reported performance accounts for turnover friction.

For BEL20, the SWARM solver runs for 14.77 s in the buy-and-hold scenario, versus 44.52 s with monthly rebalancing of portfolio weights. For OMXC20, the SWARM solver runs for 15.25 s in the buy-and-hold scenario, versus 44.02 s with monthly rebalancing of portfolio weights.

In Table 7, I present the performance comparison between buy-and-hold, initial holding, and monthly rebalancing strategies for both indices. For BEL 20, the initial holding period (months 1–24) delivers the highest mean return (1.35%) and Sharpe ratio (1.1242). The rebalancing period (months 25–60) shows negative returns (−0.06%) with decreased risk-adjusted performance (0.7584). This suggests that the portfolio benefits from stability in the first two years but faces deterioration during the rebalancing phase. For OMXC20, all strategies yield negative Sharpe ratios, indicating poor risk-adjusted performance. The holding period produces the highest returns (2.72%) but remains inadequate for positive risk compensation. The rebalancing period exhibits further degradation with returns near zero (−0.16%) and the lowest Sharpe ratio (−0.6522). The negative Sharpe ratios across OMXC20 periods suggest systematic underperformance relative to the risk-free rate.

Table 7. Rebalancing results.

BEL 20		
Monthly Period	Mean_Return	Mean_Sharpe
Buy-and-hold (1–60)	0.0050	0.9047
Holding (1–24)	0.0135	1.1242
Rebalancing (25–60)	−0.0006	0.7584
OMXC20		
Monthly Period	Mean_Return	Mean_Sharpe
Buy-and-hold (1–60)	0.0099	−0.4644
Holding (1–24)	0.0272	−0.1826
Rebalancing (25–60)	−0.0016	−0.6522

4.8.2. Out-of-Sample Forecasting

I trained the model over 5 years (January 2020–December 2024), and consider the last year 2025 as the testing window. This procedure is documented within the pseudo-code (Table A3).

To evaluate the predictive significance of the SWARM framework, I move beyond point estimates by reporting Theil’s U statistic, defined as $U = \sqrt{\frac{\sum (y_t - \hat{y}_t)^2}{\sum (y_t - y_{t-1})^2}}$, where $U < 1$ implies superior predictive power over a naive forecast. This metric assesses the model against a naive random-walk forecast; a value of $U < 1$ indicates that the algorithm successfully captures structural market signals rather than coincidental patterns. Furthermore, I utilize Equity Curve Partitioning to visually distinguish between in-sample optimization and out-of-sample validation periods.

For BEL20, the out-of-sample forecast takes 13.41 s to run. For OMXC20, the out-of-sample forecast takes 13.92 s to run.

In Table 8, I examine the generalization capability of the optimized portfolios through in-sample and out-of-sample comparison. For BEL 20, the out-of-sample period exhibits lower mean returns (0.67% vs. 1.10%) and reduced volatility (0.0344 vs. 0.0444). The Sharpe ratio improves slightly in the forecast period (1.1439 vs. 0.9824), indicating better risk-adjusted performance despite lower absolute returns. The cumulative return drops substantially from 66.13% in-sample to 8.04% out-of-sample over the shorter twelve-month horizon. For OMXC20, mean returns remain stable between periods (1.10% vs. 1.12%). Volatility decreases markedly out-of-sample (0.0530 vs. 0.0757), yet the Sharpe ratio deteriorates further (−0.5693 vs. −0.4004). This persistent negative risk-adjustment suggests the portfolio consistently underperforms the risk-free rate. The cumulative return reduction from 66.06% to 13.38% reflects both the shorter evaluation window and continued weak performance.

Table 8. Out-of-sample forecasting results.

BEL 20				
Monthly Period	Mean_Return	Volatility	Sharpe_Ratio	Cumulative_Return
In-Sample (1–60)	0.0110	0.0444	0.9824	0.6613
Out-of-Sample (61–72)	0.0067	0.0344	1.1439	0.0804
OMXC20				
Period	Mean_Return	Volatility	Sharpe_Ratio	Cumulative_Return
In-Sample (1–60)	0.0110	0.0757	−0.4004	0.6606
Out-of-Sample (61–72)	0.0112	0.0530	−0.5693	0.1338

In Table 9, I report detailed forecast accuracy and portfolio risk metrics for the out-of-sample period. For BEL 20, the MAE (2.88%) and RMSE (3.32%) indicate moderate prediction errors. The MAPE of 150% suggests high relative forecast errors, likely driven by periods where actual returns approach zero. Theil's U statistic (0.7444) below unity indicates the model outperforms naive forecasts. The Sortino ratio (1.9935) demonstrates strong downside risk-adjusted performance. Maximum drawdown remains contained at 9.51%. The win rate of 50% shows balanced positive and negative return months. For OMXC20, forecast errors are higher with MAE at 4.26% and RMSE at 5.07%. The MAPE (251.67%) and Theil's U (0.8059) confirm weaker predictive accuracy, though still superior to naive benchmarks. The negative Sortino ratio (−1.0380) reflects poor downside risk management. Maximum drawdown (7.46%) is lower than BEL 20 but occurs within an overall negative performance context. The 50% win rate indicates no directional edge in the forecast period.

Table 9. Out-of-sample forecasting metrics.

BEL 20	
Metric	Value
MAE	0.0288
RMSE	0.0332
MAPE (%)	150.0268
Theil's U	0.7444
Sortino Ratio (OOS)	1.9935
Max Drawdown (OOS)	0.0951
Win Rate (OOS)	0.5000
OMXC20	
Metric	Value
MAE	0.0426
RMSE	0.0507
MAPE (%)	251.6711
Theil's U	0.8059
Sortino Ratio (OOS)	−1.0380
Max Drawdown (OOS)	0.0746
Win Rate (OOS)	0.5000

In Figure 7, I visualize the out-of-sample forecasting performance through equity curves and Sharpe ratio comparisons for both indices. The top-left panel shows the BEL20 equity curve, which exhibits steady growth during the in-sample period (months 1–60), accumulating approximately 66% cumulative return. At month 60, marked by the red dashed line, the transition to out-of-sample validation occurs. The green segment demonstrates continued positive performance in the forecast period, adding an additional 8% despite increased volatility. The top-right panel confirms Sharpe ratio improvement from 0.98 in-sample to 1.14 out-of-sample, indicating enhanced risk-adjusted returns during validation. This improvement occurs despite lower absolute returns, reflecting the volatility reduction observed in Table 8.

The bottom-left panel presents the OMXC20 equity curve, revealing higher volatility throughout the in-sample period. The portfolio experiences significant drawdowns, notably around months 5, 32, and the terminal phase before month 60. Despite volatility, cumulative returns reach 66% by the end of in-sample optimization. The out-of-sample segment shows an initial decline followed by a recovery, accumulating 13.4% over twelve months. The bottom-right panel displays negative Sharpe ratios in both periods (−0.40 in-sample, −0.57 out-of-sample). The deterioration indicates that reduced volatility in the forecast period does not compensate for persistent underperformance relative to the risk-free rate. The equity curve's continued upward trajectory despite negative

Sharpe ratios suggests that absolute returns remain positive but fail to provide adequate risk compensation.

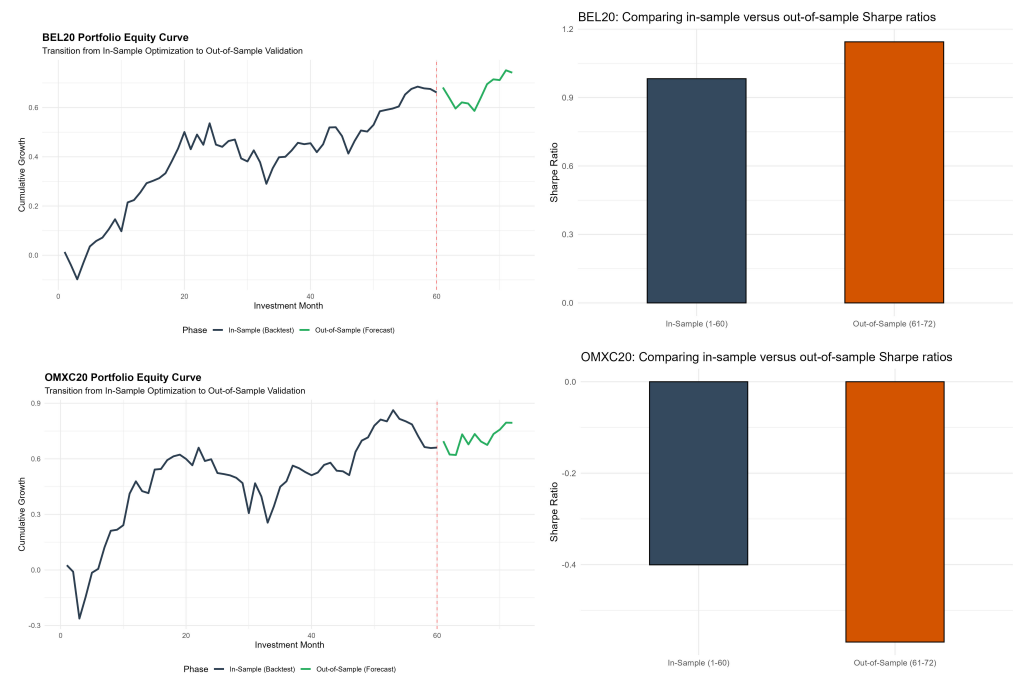


Figure 7. Out-of-sample forecasting plots for BEL20 (top) and OMXC20 (bottom). Note: The red dashed line represents the start of the out-of-sample forecasts.

4.9. Turnover

Institutional rebalancing between January 2020 and December 2025 reflected varying levels of turnover for both indices. The *BEL20* recorded the highest turnover at 20%, rotating four constituents: Proximus (PROX), Galapagos (GLPG), Aperam (APAM), and Ahold Delhaize (AHOG) were replaced by Lotus Bakeries (LOTB), Montea (MONTE), Agfa-Gevaert (AGFB), and Bekaert (BEKB). In contrast, the *OMXC20* exhibited high continuity (5% turnover), substituting only GN Store Nord (GN) with FLSmidth & Co. (FLS). These migrations underscore a pivot toward industrial logistics and defensive consumer segments, which the SWARM algorithm must navigate to maintain portfolio efficiency. In particular, the shift to Belgium's stock Lotus Bakeries (LOTB) aligns with the algorithm's preference for lower-beta defensive stocks during the volatility observed in the 2022–2023 period.

5. Conclusions

I develop an algorithmic framework that combines Particle Swarm Optimization (PSO), function stretching, and clustering to enhance portfolio optimization and asset allocation. Function stretching modifies the objective function to reduce local minima while preserving the global optimum, thereby preventing premature convergence and improving search robustness. Clustering is also incorporated to organize similar data points, enhancing both the interpretability of the solution space and the analytical capabilities of the framework. This integration yields a methodology that enhances PSO's global search capabilities, demonstrating competitive performance in complex, high-dimensional, and multi-modal optimisation, as well as in unsupervised data analysis tasks.

In terms of empirical validation, the analysis of six stock markets reveals that the effectiveness of algorithmic strategies in portfolio optimization varies by market characteristics, requiring tailored approaches. The empirical results demonstrate the generalization capability of this algorithmic family across diverse market environments,

with K-means clustering variants performing competitively in the Eurostoxx 50 and Belgian BEL 20 indices, while the baseline SWARM algorithm shows comparable efficacy in the French CAC 40 and Danish OMX Copenhagen markets. demonstrating that clustering-enhanced PSO algorithms can be systematically adapted to The key methodological contribution lies in market-specific characteristics, thereby providing portfolio managers with a comprehensive toolkit for asset allocation across different market structures. These results emphasize that the proposed algorithmic framework offers significant practical value by enabling market-specific optimization strategies, highlighting the importance of adaptive algorithms that can respond to evolving market dynamics while maintaining computational efficiency.

Quadratic programming provides greater modeling flexibility than linear programming by incorporating asset correlations in the objective function, making it ideal for Markowitz mean-variance portfolio optimization focused on risk. However, its nonlinearity presents computational challenges, as traditional gradient-descent methods often fail to find global optima, resulting in suboptimal portfolio allocations. While classical deterministic solvers excel in linear programming, they struggle with the nonconvex landscapes of constrained quadratic problems, where local minima can trap algorithms. Particle Swarm Optimization (PSO) offers a strong alternative, effectively exploring the solution space through stochastic search methods and mitigating local optima issues. This study demonstrates that PSO, especially when combined with clustering techniques and stretching functions, serves as a viable and competitive framework for large-scale quadratic portfolio optimization in real-world investment management.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in Investing.com at <https://www.investing.com/equities> (accessed on 6 January 2025).

Acknowledgments: I wish to thank the Editor, as well as two anonymous referees for their thoughtful remarks that led to improving my paper. For useful comments, I wish to thank as well Matteo Bonato, Massimiliano Mazzanti, Brian Wright, Thanasis Stengos; as well as conference participants at the CFE-CMStatistics 2024 (London, UK, December 2024), the Seoul Workshop in Empirical Finance (SWEF, Korea, May 2025), the Ferrara Economics Seminar (Italy, June 2025), and the International Conference on Risk and Financial Management (IOCRF, June 2025) on ‘Big Data, Artificial Intelligence, and Machine Learning in Finance’.

Conflicts of Interest: The author declares no conflicts of interest.

Notations

$R_{n \times m}$	Returns matrix (n periods, m stocks)
N_p	Number of particles
N_c	Number of clusters
X_i^t	Position of particle i at iteration t
V_i^t	Velocity of particle i at iteration t
f_i	Fitness of particle i
$w(t)$	Inertia weight at iteration t
$c_1(t), c_2(t)$	Cognitive/social coefficients
γ_1, γ_2, μ	Stretching params
Σ_{shrink}	Shrinkage covariance
r_f	Risk-free rate

Appendix A. Particle Swarm Optimization with Stretching and Clustering for Asset Allocation

Table A1. Comparison between PSO with stretching and traditional QP solvers.

Feature	PSO with Stretching	Traditional QP Solvers
Local Minima Handling	Escapes via function stretching	Stuck in local minima for non-convex problems
Constraints	Handles non-linear/box constraints	Limited to convex constraints
High-Dimensionality	Scalable for large portfolios ($N > 15$)	Computationally expensive ($\mathcal{O}(N^3)$)
Robustness	Tolerant to noisy covariance matrices	Sensitive to poor matrix conditioning

Table A2. Computational advantages of the integrated approach.

Component	Benefit
Hierarchical Clustering	Reduces effective dimensionality from $\mathcal{O}(n^2)$ to $\mathcal{O}(m^2)$, where $m \ll n$
Ledoit–Wolf Shrinkage	Regularizes covariance estimation for stable clustering
PSO with Stretching	Escapes local minima in non-convex cluster-constrained space

Table A3. Pseudo-code for SwarmSolver with Function Stretching.

Algorithm: SwarmSolver Optimization	
Input: Assets A , Returns matrix R , N particles, T_{max} iterations	
Output: Optimal portfolio weights $\mathbf{w}^*$	
Function SafeCovShrink(R, min_obs): $n_{assets} \leftarrow \text{ncol}(R), n_{obs} \leftarrow \text{nrow}(R)$ If min_obs is NULL then $min_obs \leftarrow \max(n_{assets} + 10, 20)$ If $n_{obs} < min_obs$ then Return Standard covariance matrix $\text{Cov}(R)$ Try: Return Ledoit–Wolf shrinkage covariance Catch error: Return Standard covariance matrix $\text{Cov}(R)$	
Initialize: Set seed $\leftarrow 233$ $w(t) \leftarrow 0.9 - (0.9 - 0.4) \cdot \frac{t}{T_{max}}$ $c_1(t) \leftarrow 2.0 - 1.5 \cdot \frac{t}{T_{max}}$ $c_2(t) \leftarrow 1.0 + 1.5 \cdot \frac{t}{T_{max}}$ $\mathbf{g}_{best} \leftarrow \mathbf{x}_1, f_{best} \leftarrow -\infty$	$\gamma_1 \leftarrow 1, \gamma_2 \leftarrow 1, \mu \leftarrow 1$ $\mathbf{x}_i \sim \mathcal{U}(0.01, 0.25)$ for $i = 1, \dots, N$ $\mathbf{v}_i \sim \mathcal{U}(-0.01, 0.01)$ for $i = 1, \dots, N$ stagnation_count $\leftarrow 0$
Function Fitness($\mathbf{w}$): $\mathbf{w} \leftarrow \frac{\mathbf{w}}{\sum \mathbf{w}}$ $\mu_p \leftarrow \mathbf{w}^T \boldsymbol{\mu}$ $\boldsymbol{\Sigma}_{shrink} \leftarrow$ Ledoit–Wolf shrinkage covariance $\sigma_p \leftarrow \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}_{shrink} \mathbf{w}}$ $r_e \leftarrow \mu_p - \bar{r}_f$ Return $\frac{r_e}{\sigma_p}$	
Function StretchFunction($f_i, f_{best}, \mathbf{x}_i, \mathbf{g}_{best}$): $G \leftarrow f_i + \gamma_1 \ \mathbf{x}_i - \mathbf{g}_{best}\ _2 \cdot (\text{sign}(f_i - f_{best}) + 1)$ $H \leftarrow G + \gamma_2 \cdot (\text{sign}(f_i - f_{best}) + 1) \cdot \tanh(\mu \cdot (G - f_{best}))$ Return H	
Function UpdateParticle(i, t): $\mathbf{r}_1, \mathbf{r}_2 \sim \mathcal{U}(0, 1)$ $\mathbf{v}_i \leftarrow w(t) \cdot \mathbf{v}_i + c_1(t) \cdot \mathbf{r}_1 \odot (\mathbf{g}_{best} - \mathbf{x}_i) + c_2(t) \cdot \mathbf{r}_2 \odot (\mathbf{g}_{best} - \mathbf{x}_i)$ $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$ $\mathbf{x}_i \leftarrow \max(0.01, \min(0.25, \mathbf{x}_i))$	

Table A3. Cont.

Algorithm: SwarmSolver Optimization
Main Optimization Loop: Set seed $\leftarrow 233$ $local_best_fitness \leftarrow -\infty$ $local_best_position \leftarrow NULL$ For $t = 1$ to T_{max} do: $stagnation_flag \leftarrow FALSE$ For $i = 1$ to N do: $current_fitness \leftarrow Fitness(x_i)$ If $current_fitness > f_{best}$ then: $f_{best} \leftarrow current_fitness$ $g_{best} \leftarrow x_i$ $stagnation_flag \leftarrow FALSE$ Else: $stagnation_flag \leftarrow TRUE$ End If End For If $stagnation_flag$ is $TRUE$ then: $stagnation_count \leftarrow stagnation_count + 1$ $\gamma_1 \leftarrow \gamma_1 + \mathcal{N}(0, 0.1)$ $\gamma_2 \leftarrow \gamma_2 + \mathcal{N}(0, 0.1)$ $\mu \leftarrow \mu + \mathcal{N}(0, 0.1)$ For $i = 1$ to N do: $stretched_fitness \leftarrow StretchFunction(current_fitness, local_best_fitness, x_i, local_best_position)$ If $stretched_fitness > local_best_fitness$ then: $local_best_fitness \leftarrow stretched_fitness$ $local_best_position \leftarrow x_i$ End If End For End If For $i = 1$ to N do: UpdateParticle(i, t) End For End For Return $w^* = \frac{g_{best}}{\sum g_{best}}$
Monthly Rebalancing Procedure: Input: Returns matrix R , holding period $T_h = 24$, rebalance range $[T_r^{start}, T_r^{end}] = [25, 60]$ Optimize on $R_{1:T_h}$ to get w_0^* Hold w_0^* for months 1 to T_h For $m = T_r^{start}$ to T_r^{end} do: Train on expanding window $R_{1:(m-1)}$ Optimize to get w_m^* Apply w_m^* to month m returns Calculate metrics: r_p^m, σ_p^m, SR^m End For Return weights history, performance metrics
Out-of-Sample Forecasting Procedure: Input: Returns matrix R , training end $T_{train} = 60$, forecast range $[T_f^{start}, T_f^{end}] = [61, 72]$ Train on $R_{1:T_{train}}$ to get w_{OOS}^* Apply w_{OOS}^* to test period $R_{T_f^{start}:T_f^{end}}$ Calculate in-sample metrics on $R_{1:T_{train}}$ Calculate out-of-sample metrics on $R_{T_f^{start}:T_f^{end}}$ Compute forecast accuracy: MAE, RMSE, MAPE, Theil's U Compute portfolio metrics: Information Ratio, Sortino, Max Drawdown, Calmar, Win Rate Return comparison statistics, forecast accuracy, additional metrics

Table A4. Pseudo-code for Hierarchical Clustering Swarm Optimization Algorithm.

Algorithm: Hierarchical Clustering and Portfolio Optimization
Input: Stock returns matrix $R \in \mathbb{R}^{T \times N}$, number of clusters K , minimum stocks per cluster M , PSO parameters Output: Optimal portfolio weights $w^* \in \mathbb{R}^N$

Table A4. Cont.

Algorithm: Hierarchical Clustering and Portfolio Optimization	
Phase 1: Hierarchical Clustering	
1. Compute distance matrix $D = \text{dist}(R^T)$ using Euclidean distance	
2. Perform hierarchical clustering $\mathcal{H} = \text{hclust}(D, \text{method} = \text{Ward.D2})$	
3. Cut dendrogram to form K clusters: $C = \text{cutree}(\mathcal{H}, k = K)$	
4. Compute cluster sizes: $ C_i $ for $i = 1, 2, \dots, K$	
5. Identify small clusters: $\mathcal{S} = \{i : C_i < M\}$	
6. for each small cluster $s \in \mathcal{S}$ do	
7. Find closest cluster c^* using hierarchical merge history	
8. Merge C_s into C_{c^*} : $C_{c^*} \leftarrow C_{c^*} \cup C_s$	
9. end for	
10. Reassign cluster labels to consecutive integers $1, 2, \dots, K'$	
11. Aggregate cluster returns: $\bar{R}_i = \frac{1}{ C_i } \sum_{j \in C_i} R_{:,j}$ for $i = 1, \dots, K'$	
Phase 2: Particle Swarm Optimization with Function Stretching	
Set seed $\leftarrow 233$	
12. Initialize PSO parameters:	
P particles, $I_{\max}$ iterations, γ_1, γ_2, μ (stretching parameters)	
13. Initialize particle positions: $X_p \sim \mathcal{U}(0.01, 0.25)^{K'}$ for $p = 1, \dots, P$	
14. Initialize velocities: $V_p \sim \mathcal{U}(-0.01, 0.01)^{K'}$ for $p = 1, \dots, P$	
15. Set global best: $g^* \leftarrow \arg \max_p f(X_p)$, $f^* \leftarrow f(g^*)$	
16. for iteration $t = 1$ to $I_{\max}$ do	
17. stagnation_flag $\leftarrow$ true	
18. for particle $p = 1$ to P do	
19. Compute fitness: $f_p^{(t)} = f(X_p^{(t)})$ where	
$f(w) = \frac{\sum_{i=1}^{K'} w_i \mu_i - r_f}{\sqrt{w^T \Sigma w}}$ (Sharpe ratio)	
20. if $f_p^{(t)} > f^*$ then	
21. $f^* \leftarrow f_p^{(t)}$, $g^* \leftarrow X_p^{(t)}$	
22. stagnation_flag $\leftarrow$ false	
23. end if	
24. end for	
25. if stagnation_flag = true then	
26. Apply adaptive parameter tuning:	
$\gamma_1 \leftarrow \gamma_1 + \mathcal{N}(0, 0.1)$	
$\gamma_2 \leftarrow \gamma_2 + \mathcal{N}(0, 0.1)$	
$\mu \leftarrow \mu + \mathcal{N}(0, 0.1)$	
27. for particle $p = 1$ to P do	
28. Compute stretched fitness:	
$G_p = f_p^{(t)} + \gamma_1 \ X_p^{(t)} - I^*\ _2 \cdot (\text{sign}(f_p^{(t)} - I_f^*) + 1)$	
$H_p = G_p + \gamma_2 \cdot (\text{sign}(f_p^{(t)} - I_f^*) + 1) \cdot \tanh(\mu(G_p - I_f^*))$	
29. if $H_p > I_f^*$ then	
30. $I_f^* \leftarrow H_p$, $I^* \leftarrow X_p^{(t)}$	
31. end if	
32. end for	
33. end if	
34. for particle $p = 1$ to P do	
35. Generate random vectors: $r_1, r_2 \sim \mathcal{U}(0, 1)^{K'}$	
36. Update velocity:	
$V_p^{(t+1)} = w(t) \cdot V_p^{(t)} + c_1(t) \cdot r_1 \odot (g^* - X_p^{(t)}) + c_2(t) \cdot r_2 \odot (g^* - X_p^{(t)})$	
37. Update position: $X_p^{(t+1)} = X_p^{(t)} + V_p^{(t+1)}$	
38. Apply bounds: $X_p^{(t+1)} = \max(0.01, \min(0.25, X_p^{(t+1)}))$	
39. end for	
40. end for	
41. Normalize cluster weights: $w_c^* = \frac{g^*}{\sum_{i=1}^{K'} g_i^*}$	
Phase 3: Weight Distribution to Individual Stocks	
42. Initialize individual stock weights: $w_s = 0$ for $s = 1, \dots, N$	
43. for cluster $i = 1$ to K' do	
44. Compute equal weight per stock: $w_{s,i} = \frac{w_{c,i}^*}{ C_i }$	
45. for stock $s \in C_i$ do	
46. $w_s = w_{s,i}$	
47. end for	
48. end for	
49. Return $w^* = (w_1, w_2, \dots, w_N)$	

Table A4. Cont.

Algorithm: Hierarchical Clustering and Portfolio Optimization
Parameter Functions: $w(t) = w_{\max} - (w_{\max} - w_{\min}) \cdot \frac{t}{T_{\max}}$ where $w_{\max} = 0.9, w_{\min} = 0.4$ $c_1(t) = 2.0 - 1.5 \cdot \frac{t}{T_{\max}}$ (cognitive coefficient) $c_2(t) = 1.0 + 1.5 \cdot \frac{t}{T_{\max}}$ (social coefficient)
Additional Functions: Fitness Function: $f(w) = \frac{\sum_{i=1}^{K'} w_i \mu_i - r_f}{\sqrt{w^T \Sigma w}}$ where μ_i is the mean return of cluster i , r_f is the risk-free rate, and Σ is the shrinkage covariance matrix Stretching Functions: $G_p = f_p^{(t)} + \gamma_1 \ X_p^{(t)} - l^*\ _2 \cdot (\text{sign}(f_p^{(t)} - l_f^*) + 1)$ $H_p = G_p + \gamma_2 \cdot (\text{sign}(f_p^{(t)} - l_f^*) + 1) \cdot \tanh(\mu(G_p - l_f^*))$ where l^* is the local best position and l_f^* is the local best fitness
Algorithm Complexity: Hierarchical clustering: $O(N^3)$ PSO optimization: $O(P \cdot I_{\max} \cdot K')$ Overall complexity: $O(N^3 + P \cdot I_{\max} \cdot K')$

Table A5. Pseudo-code for K-means Clustering with Particle Swarm Optimization.

Algorithm: K-Means Clustering with Enhanced Particle Swarm Optimization
Input: Stock returns matrix R , number of clusters k , number of particles n , maximum iterations T Output: Optimal portfolio weights w^*
Phase 1: K-means Clustering 1. for $j = 2$ to 10 do 2. Perform K-means clustering on R^T with j clusters 3. Calculate silhouette score S_j 4. end for 5. Select optimal number of clusters $k^* = \arg \max_j S_j$ 6. repeat 7. Apply K-means clustering with k^* centers 8. Assign stocks to clusters: $C_i = \{s : s \in \text{cluster } i\}$ 9. until all clusters have $ C_i \geq 2$ 10. Aggregate returns by cluster: $\bar{R}_i = \frac{1}{ C_i } \sum_{s \in C_i} R_s$
Phase 2: Enhanced Particle Swarm Optimization Set seed $\leftarrow 233$ 11. Initialize particles: $X_i \sim \mathcal{U}(0.01, 0.25)^k$ for $i = 1, \dots, n$ 12. Initialize velocities: $V_i \sim \mathcal{U}(-0.01, 0.01)^k$ for $i = 1, \dots, n$ 13. Set stretching parameters: $\gamma_1 = 1, \gamma_2 = 1, \mu = 1$ 14. Initialize global best: $g^* = \arg \max_i f(X_i)$ 15. for $t = 1$ to T do 16. $\text{stagnation_flag} = \text{true}$ 17. for $i = 1$ to n do 18. Calculate fitness: $f_i = f(X_i) = \frac{\mathbb{E}[r_p] - r_f}{\sqrt{\text{Var}(r_p)}}$ 19. if $f_i > f(g^*)$ then 20. $g^* = X_i, \text{stagnation_flag} = \text{false}$ 21. end if 22. end for 23. if $\text{stagnation_flag} = \text{true}$ then 24. Adaptive parameter tuning: 25. $\gamma_1 = \gamma_1 + \mathcal{N}(0, 0.1)$ 26. $\gamma_2 = \gamma_2 + \mathcal{N}(0, 0.1)$ 27. $\mu = \mu + \mathcal{N}(0, 0.1)$ 28. Apply stretching transformation: 29. $G_i = f_i + \gamma_1 \ X_i - g^*\ _2 \cdot (\text{sign}(f_i - f(g^*)) + 1)$ 30. $H_i = G_i + \gamma_2 \cdot (\text{sign}(f_i - f(g^*)) + 1) \cdot \tanh(\mu(G_i - f(g^*)))$ 31. end if 32. for $i = 1$ to n do 33. Update inertia: $w = 0.9 - 0.5 \cdot \frac{t}{T}$ 34. Update coefficients: $c_1 = 2.0 - 1.5 \cdot \frac{t}{T}, c_2 = 1.0 + 1.5 \cdot \frac{t}{T}$ 35. Generate random vectors: $r_1, r_2 \sim \mathcal{U}(0, 1)^k$ 36. Update velocity: $V_i = w \cdot V_i + c_1 \cdot r_1 \circ (g^* - X_i) + c_2 \cdot r_2 \circ (g^* - X_i)$ 37. Update position: $X_i = X_i + V_i$

Table A5. Cont.

Algorithm: K-Means Clustering with Enhanced Particle Swarm Optimization	
38.	Apply bounds: $X_i = \max(0.01, \min(0.25, X_i))$
39.	end for
40.	end for
41.	Normalize weights: $w^* = \frac{g^*}{\sum_{j=1}^k g_j^*}$
Phase 3: Weight Distribution to Individual Stocks	
42.	Initialize individual stock weights: $w_{stock} = \mathbf{0}^{ S }$ where $ S $ is total number of stocks
43.	for $i = 1$ to k do
44.	Get stocks in cluster i : $S_i = \{s : s \in C_i\}$
45.	Get optimized cluster weight: $w_{cluster}^{(i)} = w_i^*$
46.	Calculate weight per stock: $w_{per_stock}^{(i)} = \frac{w_{cluster}^{(i)}}{ S_i }$
47.	for each stock $s \in S_i$ do
48.	Assign weight: $w_{stock}[s] = w_{per_stock}^{(i)}$
49.	end for
50.	end for
51.	Create final portfolio: $\mathcal{P} = \{(s, C_s, w_{stock}[s]) : s \in S\}$
52.	return $\mathcal{P}$
Fitness Function:	
$f(w) = \frac{w^T \mu - r_f}{\sqrt{w^T \Sigma w}}$	
where μ is the expected return vector, Σ is the shrinkage covariance matrix, and r_f is the risk-free rate.	

Table A6. DBSCAN Clustering and PSO Initialization with Adaptive Stretching.

Algorithm: DBSCAN-PSO Portfolio Optimization	
1	Input: Returns matrix $R_{n \times m}$ where n = time periods, m = stocks
2	Data Scaling: $R_{scaled} = \text{standardize}(R)$
3	Estimate ϵ parameter:
4	Compute k -nearest neighbor distances: $d_k = \text{kNNdist}(R_{scaled}^T, k = 4)$
5	Sort distances: $d_{sorted} = \text{sort}(d_k)$
6	Set initial $\epsilon = \text{median}(d_{sorted}) \times 1.2$
7	Adaptive DBSCAN:
8	Initialize: minPts = 5, target_clusters = 7, max_iter = 20
9	iter = 1
10	while found_clusters < target_clusters and iter $\leq$ max_iter:
11	$C = \text{DBSCAN}(R_{scaled}^T, \epsilon, \text{minPts})$
12	found_clusters = $ \text{unique}(C) - 1$ (exclude noise)
13	if found_clusters ≤ 1 :
14	$\epsilon = \max(\epsilon \times 0.85, \text{min_eps})$
15	end if
16	iter = iter + 1
17	end while
18	Cluster Aggregation:
19	for each cluster i :
20	$R_i = \text{mean}(R_{scaled}[\text{stocks in cluster } i])$
21	end for
22	Output: Clustered returns matrix $R_{clustered}$ Set seed $\leftarrow 233$
23	PSO Initialization:
24	for $i = 1$ to N_p :
25	for $j = 1$ to N_c :
26	$X_{i,j} \sim \text{Uniform}(0.01, 0.25)$
27	$V_{i,j} \sim \text{Uniform}(-0.01, 0.01)$
28	end for
29	end for
30	Parameters: $\gamma_1 = 1, \gamma_2 = 1, \mu = 1$
31	$w_{\max} = 0.9, w_{\min} = 0.1$
32	PSO Main Loop:
33	for $t = 1$ to max_iter:
34	stagnation_flag = True
35	for $i = 1$ to N_p :

Table A6. Cont.

Algorithm:	DBSCAN-PSO Portfolio Optimization
36	Normalize weights: $w_i = \frac{X_i}{\sum_j X_{ij}}$
37	Portfolio return: $r_p = \sum_j w_j \cdot \bar{r}_j$
38	Shrinkage covariance: $\Sigma_{shrink} = \text{cov.shrink}(R_{clustered})$
39	Portfolio risk: $\sigma_p = \sqrt{w^T \Sigma_{shrink} w}$
40	Fitness (Sharpe ratio): $f_i^t = \frac{r_p - r_f}{\sigma_p}$
41	if $f_i^t > f_{best}^t$:
42	$f_{best} = f_i^t$, $X_{best} = X_i^t$
43	stagnation_flag = False
44	end if
45	end for
46	Adaptive Parameters:
47	$w(\text{iter}) = w_{\max} - (w_{\max} - w_{\min}) \cdot \frac{t}{\max_iter}$
48	$c_1(t) = 2.0 - 1.5 \cdot \frac{t}{\max_iter}$
49	$c_2(t) = 1.0 + 1.5 \cdot \frac{t}{\max_iter}$
50	if stagnation_flag = True:
51	stagnation_count = stagnation_count + 1
52	$\gamma_1 = \gamma_1 + \mathcal{N}(0, 0.1)$
53	$\gamma_2 = \gamma_2 + \mathcal{N}(0, 0.1)$
54	$\mu = \mu + \mathcal{N}(0, 0.1)$
55	Apply Stretching Function:
56	for $i = 1$ to N_p :
57	$G = f_i + \gamma_1 \cdot \ X_i - X_{best}^{local}\ _2 \cdot (\text{sign}(f_i - f_{best}^{local}) + 1)$
58	$H = G + \gamma_2 \cdot (\text{sign}(f_i - f_{best}^{local}) + 1) \cdot \tanh(\mu \cdot (G - f_{best}^{local}))$
59	$f_i^{stretched} = H$
60	if $f_i^{stretched} > f_{best}^{local}$:
61	$f_{best}^{local} = f_i^{stretched}$, $X_{best}^{local} = X_i^t$
62	end if
63	end for
64	end if
65	Update Particles:
66	for $i = 1$ to N_p :
67	for $j = 1$ to N_c :
68	$r_1, r_2 \sim \text{Uniform}(0, 1)$
69	$V_{ij}^{t+1} = w(t) \cdot V_{ij}^t + c_1(t) \cdot r_1 \cdot (X_{best,j} - X_{ij}^t) + c_2(t) \cdot r_2 \cdot (X_{best,j} - X_{ij}^t)$
70	$X_{ij}^{t+1} = X_{ij}^t + V_{ij}^{t+1}$
71	$X_{ij}^{t+1} = \max(0.01, \min(0.25, X_{ij}^{t+1}))$
72	end for
73	end for
74	end for
75	Weight Distribution to Individual Stocks:
76	Optimal cluster weights: $w_{cluster} = \frac{X_{best}}{\sum_j X_{best,j}}$
77	Initialize: $w_{stock} = [0, 0, \dots, 0]$ (length m)
78	$U = \text{unique}(C)$
79	for each cluster $k \in U$:
80	$I_k = \{i : c_i = k\}$
81	$n_k = I_k $
82	if $n_k > 0$ and $k + 1 \leq w_{cluster} $:
83	$w_k = w_{cluster}[k + 1]$
84	$w_{per_stock} = \frac{w_k}{n_k}$
85	for each stock $i \in I_k$:
86	$w_{stock}[i] = w_{per_stock}$
87	end for
88	end if
89	end for
90	Validation and Output:
91	Check: $\sum_{i=1}^m w_{stock}[i] \approx 1$
92	Expected return: $E[r_p] = \sum_{i=1}^m w_{stock}[i] \cdot E[r_i]$
93	Portfolio risk: $\sigma_p = \sqrt{w_{stock}^T \Sigma w_{stock}}$
94	Return optimal portfolio weights w_{stock}

Table A7. Portfolio Weights for DJ Euro Stoxx 50.

Stock	Cluster	Weight	Percentage	Rank
Adyen	5	0.0673	6.7313	1
Hermes	5	0.0673	6.7313	2
Ferrari	5	0.0673	6.7313	3
Scheider Electric	5	0.0673	6.7313	4
ASML Holding	5	0.0673	6.7313	5
Dassault Systemes	5	0.0673	6.7313	6
Enel	7	0.0449	4.4875	7
Deutsche Boerse	7	0.0449	4.4875	8
Danone	7	0.0449	4.4875	9
Air Liquide	7	0.0449	4.4875	10
Deutsche Telekom AG	7	0.0449	4.4875	11
SAP	7	0.0449	4.4875	12
Ahold Delhaize	7	0.0449	4.4875	13
SANOFI	7	0.0449	4.4875	14
IBERDROLA	7	0.0449	4.4875	15
Safran	6	0.0178	1.7843	16
AIRBUS	6	0.0178	1.7843	17
BBVA	6	0.0178	1.7843	18
SANTANDER	6	0.0178	1.7843	19
BASF	4	0.0060	0.6034	20
Bayer	4	0.0060	0.6034	21
Allianz	4	0.0060	0.6034	22
Eni	4	0.0060	0.6034	23
Anheuser Busch	4	0.0060	0.6034	24
EssilorLuxottica	4	0.0060	0.6034	25
AXA	4	0.0060	0.6034	26
Munich Re Group	4	0.0060	0.6034	27
TOTAL ENERGIES	4	0.0060	0.6034	28
VINCI	4	0.0060	0.6034	29
INDITEX	4	0.0060	0.6034	30
MICHELIN	4	0.0060	0.6034	31
ING	3	0.0054	0.5385	32
Intensa	3	0.0054	0.5385	33
BNP Paribas	3	0.0054	0.5385	34
Volkswagen	1	0.0023	0.2308	35
Stellantis	1	0.0023	0.2308	36
Siemens	1	0.0023	0.2308	37
Mercedes	1	0.0023	0.2308	38
BMW ST	1	0.0023	0.2308	39
Infineon	1	0.0023	0.2308	40
Flutter Entertainment	1	0.0023	0.2308	41
Philips	2	0.0018	0.1795	42
Deutshce Post	2	0.0018	0.1795	43
Adidas	2	0.0018	0.1795	44
Prosus	2	0.0018	0.1795	45
Pernod Ricard	2	0.0018	0.1795	46
L'Oréal	2	0.0018	0.1795	47
Louis Vuitton	2	0.0018	0.1795	48
Kering	2	0.0018	0.1795	49
KONE	2	0.0018	0.1795	50

Table A8. Portfolio Weights for Nifty 50.

Stock	Cluster	Weight	Percentage	Rank
Cipla Ltd.	4	0.0717	7.1731	1
Dr. Reddy's Laboratories Ltd.	4	0.0717	7.1731	2
HCL Technologies Ltd.	4	0.0717	7.1731	3
Bharat Petroleum Corporation Ltd.	4	0.0717	7.1731	4
Britannia Industries Ltd.	4	0.0717	7.1731	5
Hindalco Industries Ltd.	4	0.0717	7.1731	6
Tata Motors Ltd.	4	0.0717	7.1731	7
Wipro Ltd.	4	0.0717	7.1731	8
Axis Bank Ltd.	3	0.0141	1.4083	9
Bajaj Auto Ltd.	3	0.0141	1.4083	10
Bharat Petroleum Corporation Ltd.	3	0.0141	1.4083	11
Britannia Industries Ltd.	3	0.0141	1.4083	12
Bajaj Auto Ltd.	3	0.0141	1.4083	13
Coal India Ltd.	3	0.0141	1.4083	14
Eicher Motors Ltd.	3	0.0141	1.4083	15
Grasim Industries Ltd.	3	0.0141	1.4083	16
HDFC Bank Ltd.	3	0.0141	1.4083	17
Hero MotoCorp Ltd.	3	0.0141	1.4083	18
ICICI Bank Ltd.	3	0.0141	1.4083	19
ITC Ltd.	3	0.0141	1.4083	20
Kotak Mahindra Bank Ltd.	3	0.0141	1.4083	21
Larsen & Toubro Ltd.	3	0.0141	1.4083	22
Mahindra & Mahindra Ltd.	3	0.0141	1.4083	23
Maruti Suzuki India Ltd.	3	0.0141	1.4083	24
Coal India Ltd.	3	0.0141	1.4083	25
Eicher Motors Ltd.	3	0.0141	1.4083	26
Grasim Industries Ltd.	3	0.0141	1.4083	27
HDFC Bank Ltd.	3	0.0141	1.4083	28
Hero MotoCorp Ltd.	3	0.0141	1.4083	29
Tata Motors Ltd.	3	0.0141	1.4083	30
Titan Company Ltd.	3	0.0141	1.4083	31
Trent Ltd.	3	0.0141	1.4083	32
UltraTech Cement Ltd.	3	0.0141	1.4083	33
SBI Life Insurance Co., Ltd.	3	0.0141	1.4083	34
HDFC Life Insurance Co., Ltd.	3	0.0141	1.4083	35
Aspinwall and Company Ltd.	2	0.0038	0.3826	36
Bharti Airtel Ltd.	2	0.0038	0.3826	37
Hindustan Unilever Ltd.	2	0.0038	0.3826	38
IndusInd Bank Ltd.	2	0.0038	0.3826	39
Infosys Ltd.	2	0.0038	0.3826	40
Nestle India Ltd.	2	0.0038	0.3826	41
Adani Enterprises Ltd.	1	0.0026	0.2550	42
APL Apollo Tubes Ltd.	1	0.0026	0.2550	43
Bajaj Finance Ltd.	1	0.0026	0.2550	44
Bajaj Finserv Ltd.	1	0.0026	0.2550	45
Hindalco Industries Ltd.	1	0.0026	0.2550	46
IndusInd Bank Ltd.	1	0.0026	0.2550	47
Infosys Ltd.	1	0.0026	0.2550	48
JSW Steel Ltd.	1	0.0026	0.2550	49
Tata Steel Ltd.	1	0.0026	0.2550	50

Table A9. Portfolio Weights for China A50.

Stock	Cluster	Weight	Percentage	Rank
Jiangsu Hengrui	4	0.0958	9.9206	1
Kweichow Moutai	4	0.0958	9.9206	2
Inner Mongolia Yili	4	0.0958	9.9206	3
Yanghe Brewery A	4	0.0958	9.9206	4
Foshan Haitian Food	4	0.0958	9.9206	5
Hik Vision Digital A	4	0.0958	9.9206	6
S.F. Holding Co.	4	0.0958	9.9206	7
Wuliangye A	4	0.0958	9.9206	8
Shenzhen Mindray Bio-Medical	4	0.0958	9.9206	9
Shanghai International Port	3	0.0115	1.1905	10
360 Security Technology	3	0.0115	1.1905	11
Foxconn Industrial Internet	3	0.0115	1.1905	12
China Shenhua Energy SH	1	0.0029	0.2976	13
Bank of China A	1	0.0029	0.2976	14
China Petrol A	1	0.0029	0.2976	15
China Minsheng Banking	1	0.0029	0.2976	16
China Construction Bank Co.	1	0.0029	0.2976	17
China Yangtze Power	1	0.0029	0.2976	18
Bank of Beijing	1	0.0029	0.2976	19
China Citic Bank A	1	0.0029	0.2976	20
Agricultural Bank China A	1	0.0029	0.2976	21
PetroChina A	1	0.0029	0.2976	22
Bank of Communications Co., Ltd.	1	0.0029	0.2976	23
ICBC	1	0.0029	0.2976	24
Pudong Development Bank	2	0.0017	0.1786	25
CITIC Securities	2	0.0017	0.1786	26
China Merchants Bank	2	0.0017	0.1786	27
China United Network Comm	2	0.0017	0.1786	28
Poly Real Estate Group	2	0.0017	0.1786	29
SAIC Motor Corp	2	0.0017	0.1786	30
Anhui Conch Cement	2	0.0017	0.1786	31
Industrial Bank	2	0.0017	0.1786	32
New China Life Insurance	2	0.0017	0.1786	33
Ping An Insurance	2	0.0017	0.1786	34
China Pacific Insurance	2	0.0017	0.1786	35
China Life Insurance A	2	0.0017	0.1786	36
China Everbright Bank	2	0.0017	0.1786	37
China Vanke A	2	0.0017	0.1786	38
Gree Electric A	2	0.0017	0.1786	39
Midea Group A	2	0.0017	0.1786	40
Ping An Bank A	2	0.0017	0.1786	41
Guotai Junan Securities	2	0.0017	0.1786	42
China Merchants Shekou	2	0.0017	0.1786	43
Guangdong Wens Foodstuff	2	0.0017	0.1786	44
China Railway Construction	5	0.0000	0.0000	45
China Railway A	5	0.0000	0.0000	46
China State Construction	5	0.0000	0.0000	47
China Communications Construction	5	0.0000	0.0000	48
CRRC A	5	0.0000	0.0000	49
Baoshan Iron & Steel	5	0.0000	0.0000	50

Table A10. Portfolio Weights for CAC 40.

Stock	Weight	Percentage	Rank
TotalEnergies SE	0.0504	5.0403	1
Air Liquide	0.0504	5.0403	2
Stellantis NV	0.0504	5.0403	3
ArcelorMittal	0.0504	5.0403	4
Vinci SA	0.0504	5.0403	5
AXA	0.0504	5.0403	6
BNP Paribas	0.0504	5.0403	7
Airbus SE	0.0504	5.0403	8
Legrand	0.0504	5.0403	9
Crédit Agricole	0.0504	5.0403	10
EssilorLuxottica	0.0504	5.0403	11
Schneider Electric	0.0504	5.0403	12
Publicis Groupe	0.0504	5.0403	13
Société Générale	0.0504	5.0403	14
Safran	0.0504	5.0403	15
Saint-Gobain SA	0.0504	5.0403	16
Sanofi	0.0504	5.0403	17
Renault	0.0504	5.0403	18
Unibail-Rodamco-Westfield	0.0504	5.0403	19
LVMH	0.0020	0.2016	20
Bouygues	0.0020	0.2016	21
Accor	0.0020	0.2016	22
Dassault Systèmes	0.0020	0.2016	23
Eurofins Scientific	0.0020	0.2016	24
Engie	0.0020	0.2016	25
Capgemini	0.0020	0.2016	26
Carrefour	0.0020	0.2016	27
Hermès International	0.0020	0.2016	28
Danone	0.0020	0.2016	29
Michelin	0.0020	0.2016	30
Teleperformance	0.0020	0.2016	31
Edenred	0.0020	0.2016	32
Pernod Ricard	0.0020	0.2016	33
Veolia	0.0020	0.2016	34
STMicroelectronics	0.0020	0.2016	35
Pernod Ricard	0.0020	0.2016	36
Orange	0.0020	0.2016	37
Vivendi	0.0020	0.2016	38
Alstom	0.0020	0.2016	39
Thales	0.0020	0.2016	40

Table A11. Portfolio Weights for BEL 20.

Stock	Cluster	Weight	Percentage	Rank
arGEN-X SE	4	0.2683	26.8319	1
Umicore	1	0.0448	4.4833	2
Solvay	1	0.0448	4.4833	3
Proximus	1	0.0448	4.4833	4
Melexis NV	1	0.0448	4.4833	5
KBC Group	1	0.0448	4.4833	6
Aperam	1	0.0448	4.4833	7
Anheuser-Busch InBev	1	0.0448	4.4833	8
Ageas	1	0.0448	4.4833	9
WDP	2	0.0359	3.5867	10

Table A11. *Cont.*

Stock	Cluster	Weight	Percentage	Rank
UCB	2	0.0359	3.5867	11
Sofina	2	0.0359	3.5867	12
D'Ieteren Group SA	2	0.0359	3.5867	13
GBL	2	0.0359	3.5867	14
Elia System Operator	2	0.0359	3.5867	15
Cofinimmo	2	0.0359	3.5867	16
Aedifica NV	2	0.0359	3.5867	17
Ackermans & van Haaren	2	0.0359	3.5867	18
Koninklijke Ahold Delhaize N.V.	2	0.0359	3.5867	19
Galapagos NV	3	0.0143	1.4347	20

Table A12. Portfolio Weights for OMX C20.

Stock	Weight	Percentage	Rank
NKT A/S	0.1786	17.8571	1
Vestas Wind Systems A/S	0.1786	17.8571	2
Jyske Bank A/S	0.1786	17.8571	3
Pandora A/S	0.1786	17.8571	4
Rockwool A/S	0.1786	17.8571	5
Danske Bank A/S	0.0071	0.7143	6
Carlsberg Group A/S (class B)	0.0071	0.7143	7
DSV A/S	0.0071	0.7143	8
Novo Nordisk A/S (class B)	0.0071	0.7143	9
A.P. Moller-Maersk A/S (class B)	0.0071	0.7143	10
Novonesis B (formerly Novozymes)	0.0071	0.7143	11
A.P. Moller-Maersk A/S (class A)	0.0071	0.7143	12
Genmab A/S	0.0071	0.7143	13
GN Store Nord A/S	0.0071	0.7143	14
Demant A/S	0.0071	0.7143	15
Tryg A/S	0.0071	0.7143	16
Coloplast A/S (class B)	0.0071	0.7143	17
Zealand Pharma A/S	0.0071	0.7143	18
ISS A/S	0.0071	0.7143	19
Orsted A/S	0.0071	0.7143	20

Note

¹ i.e., assuming 3 basis points per transaction side.

References

- Abubaker, A., Baharum, A., & Alrefaei, M. (2015). Automatic clustering using multi-objective particle swarm and simulated annealing. *PLoS ONE*, 10(7), e0130995. [\[CrossRef\]](#)
- Aguiar Nascimento, R., Neto, Á. B., Bezerra, Y. S. d. F., do Nascimento, H. A. D., Lucena, L. d. S., & de Freitas, J. E. (2022). A new hybrid optimization approach using pso, nelder-mead simplex and kmeans clustering algorithms for 1d full waveform inversion. *PLoS ONE*, 17(12), e0277900. [\[CrossRef\]](#) [\[PubMed\]](#)
- Ananthi, M., Valarmathi, K., Ramathilagam, A., & Praveen, R. (2025). Hybrid lion and exponential pso-based metaheuristic clustering approach for efficient dynamic data stream management. *Nature Scientific Reports*, 15(1), 22343. [\[CrossRef\]](#)
- Bulani, V., Bezbradica, M., & Crane, M. (2025). Improving portfolio management using clustering and particle swarm optimisation. *Mathematics*, 13(10), 1623. [\[CrossRef\]](#)
- Chen, R.-R., Huang, W. K., & Yeh, S.-K. (2021). Particle swarm optimization approach to portfolio construction. *Intelligent Systems in Accounting, Finance and Management*, 28(3), 182–194. [\[CrossRef\]](#)
- Eberhart, R., & Kennedy, J. (1995, November 27–December 1). *Particle swarm optimization*. IEEE International Conference on Neural Networks (pp. 1942–1948), Perth, Australia.
- Erwin, K., & Engelbrecht, A. (2023). Meta-heuristics for portfolio optimization. *Soft Computing*, 27(24), 19045–19073. [\[CrossRef\]](#)

- Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996, August 2). *A density-based algorithm for discovering clusters in large spatial databases with noise*. 2nd International Conference on Knowledge Discovery and Data Mining (pp. 226–231), Portland, OR, USA.
- Freitas, D., Lopes, L. G., & Morgado-Dias, F. (2020). Particle swarm optimisation: A historical review up to the current developments. *Entropy*, 22(3), 362. [\[CrossRef\]](#) [\[PubMed\]](#)
- Hartigan, J. A., & Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1), 100–108. [\[CrossRef\]](#)
- Hayashida, T., Sekizaki, S., Shimoyoshi, K., & Nishizaki, I. (2025). ELPSO-C: A clustering-based strategy for dimension-wise diversity control in enhanced leader particle swarm optimization. *AppliedMath*, 5(4), 159. [\[CrossRef\]](#)
- Huang, K. Y. (2011). A hybrid particle swarm optimization approach for clustering and classification of datasets. *Knowledge-Based Systems*, 24(3), 420–426. [\[CrossRef\]](#)
- Jiang, Z., Zhu, D., Li, X.-Y., & Han, L.-B. (2025). A hybrid adaptive particle swarm optimization algorithm for enhanced performance. *Applied Sciences*, 15(11), 6030. [\[CrossRef\]](#)
- Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3), 241–254. [\[CrossRef\]](#)
- Khan, K., Rehman, S. U., Aziz, K., Fong, S., & Sarasvady, S. (2014, February 17–19). *Dbscan: Past, present and future*. Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014) (pp. 232–238), Chennai, India.
- Ledoit, O., & Wolf, M. (2003). Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, 10(5), 603–621. [\[CrossRef\]](#)
- Li, G., Wang, W., Zhang, W., Wang, Z., Tu, H., & You, W. (2021). Grid search based multi-population particle swarm optimization algorithm for multimodal multi-objective optimization. *Swarm and Evolutionary Computation*, 62, 100843. [\[CrossRef\]](#)
- Likas, A., Vlassis, N., & Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern Recognition*, 36(2), 451–461. [\[CrossRef\]](#)
- Lin, K., Cai, T., Wang, H., Li, W., & Wei, C. (2025). Integrated learning particle swarm optimization algorithm based on clustering. *International Journal of Cognitive Informatics & Natural Intelligence*, 19(1), 1–24.
- Lolic, M. (2024). Practical improvements to mean-variance optimization for multi-asset class portfolios. *Journal of Risk and Financial Management*, 17(5), 183. [\[CrossRef\]](#)
- Markowitz, H. M. (1952). Portfolio selection. *Journal of Finance*, 7(1), 71–91.
- Muteba Mwamba, J. W., Mbucici, L. M., & Mba, J. C. (2025). Multi-objective portfolio optimization: An application of the non-dominated sorting genetic algorithm iii. *International Journal of Financial Studies*, 13(1), 15. [\[CrossRef\]](#)
- Niknam, T., & Amiri, B. (2010). An efficient hybrid approach based on pso, aco and k-means for cluster analysis. *Applied Soft Computing*, 10(1), 183–197. [\[CrossRef\]](#)
- Niu, Y., Yan, X., Zeng, W., Wang, Y., & Niu, Y. (2025). Multi-objective sand cat swarm optimization based on adaptive clustering for solving multimodal multi-objective optimization problems. *Mathematics and Computers in Simulation*, 227, 391–404. [\[CrossRef\]](#)
- Ntare, H. B., Muteba Mwamba, J. W., & Adekambi, F. (2025). Dynamic portfolio optimization with diversification analysis and asset selection amidst high correlation using cryptocurrencies and bank equities. *Risks*, 13(6), 113. [\[CrossRef\]](#)
- Parsopoulos, K. E., Plagianakos, V. P., Magoulas, G. D., & Vrahatis, M. N. (2001a). Improving the particle swarm optimizer by function “stretching”. In *Advances in convex analysis and global optimization: Honoring the memory of C. caratheodory (1873–1950)* (pp. 445–457). Springer.
- Parsopoulos, K. E., Plagianakos, V. P., Magoulas, G. D., & Vrahatis, M. N. (2001b). Objective function “stretching” to alleviate convergence to local minima. *Nonlinear Analysis-Theory Methods and Applications*, 47(5), 3419–3424. [\[CrossRef\]](#)
- Parsopoulos, K. E., & Vrahatis, M. N. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2), 235–306. [\[CrossRef\]](#)
- Parsopoulos, K. E., & Vrahatis, M. N. (2004). On the computation of all global minimizers through particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3), 211–224. [\[CrossRef\]](#)
- Passaro, A., & Starita, A. (2008). Particle swarm optimization for multimodal functions: A clustering approach. *Journal of Artificial Evolution and Applications*, 2008(1), 482032. [\[CrossRef\]](#)
- Sengupta, S., Basak, S., & Peters, R. A. (2018). Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowledge Extraction*, 1(1), 157–191. [\[CrossRef\]](#)
- Verma, H., Verma, D., & Tiwari, P. K. (2021). A population based hybrid fcm-pso algorithm for clustering analysis and segmentation of brain image. *Expert Systems with Applications*, 167, 114121. [\[CrossRef\]](#)
- Yao, J., Luo, X., Li, F., Li, J., Dou, J., & Luo, H. (2024). Research on hybrid strategy particle swarm optimization algorithm and its applications. *Nature Scientific Reports*, 14(1), 24928. [\[CrossRef\]](#)
- Yuan, Y., Yang, P., Jiang, H., & Shi, T. (2024). A multi-robot task allocation method based on the synergy of the K-Means++ algorithm and the particle swarm algorithm. *Biomimetics*, 9(11), 694.

- Zhan, Z.-H., Zhang, J., Li, Y., & Chung, H. S.-H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6), 1362–1381. [[CrossRef](#)] [[PubMed](#)]
- Zhang, R.-L., & Liu, X.-H. (2023). A novel hybrid high-dimensional pso clustering algorithm based on the cloud model and entropy. *Applied Sciences*, 13(3), 1246. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.