ELSEVIER

A hybrid level-based learning swarm algorithm with mutation operator for solving large-scale cardinality-constrained portfolio optimization problems

Massimiliano Kaucic *, Filippo Piccotto, Gabriele Sbaiz, Giorgio Valentinuz

University of Trieste, Department of Economics, Business, Mathematics and Statistics, Via Valerio 4/1, 34127 Trieste, Italy

A R T I C L E   I N F O

A B S T R A C T

We propose a hybrid variant of the level-based learning swarm optimizer (LLSO) for solving large-scale portfolio optimization problems. This solver fills the gap due to the inadequacy of the particle swarm optimization algorithm for high-dimensional instances. We aim to extend the classical mean-variance formulation by maximizing a modified version of the Sharpe ratio subject to cardinality, box, and budget constraints. The algorithm involves a projection operator to deal with these three constraints simultaneously. Further, we implicitly control transaction costs thanks to a rebalancing constraint handled by a suitable exact penalty function. In addition, we develop an ad hoc mutation operator to modify candidate exemplars in the highest level of the swarm. The experimental results, using three large-scale data sets, show that including this procedure improves the accuracy of the solutions. Then, a comparison with other variants of the LLSO algorithm and two state-of-the-art swarm optimizers points out the outstanding performance of the proposed solver in terms of exploration capabilities and solution quality. Finally, we assess the profitability of the portfolio allocation strategy in the last five years using an investable pool of 1119 constituents from the MSCI World Index.

1. Introduction

The mean-variance portfolio selection problem developed by Markowitz underpins the modern portfolio theory. According to this approach, the portfolio returns distribution's mean and variance represents the investment profit and the risk, respectively. A portfolio is efficient if it provides the maximum return for a given level of risk or, equivalently, if it has the minimum risk for a given level of return. Therefore, the efficient frontier is provided by the set of optimal mean-variance trade-offs in the risk-return space.

In recent years, many developments of the basic model have been investigated (see, for instance, [19]). On the one hand, researchers have defined new risk measures to describe better investor attitudes [24]. On the other hand, the issues linked to portfolio management have been addressed by introducing several constraints to handle portfolio weights (a detailed list is given in [27]). In this context, some authors have studied the multi-objective formulation of the mean-variance portfolio optimization problem, in which the expected portfolio return is maximized and, at the same time, its variance is minimized (see, for instance, [22]). The scope is to provide heuristics able to generate accurate dotted representations of the set of efficient portfolios in a few iterations.

---

* Corresponding author.

E-mail addresses: massimiliano.kaucic@deams.units.it (M. Kaucic), filippo.piccotto@phd.units.it (F. Piccotto), gabriele.sbaiz@phd.units.it (G. Sbaiz), giorgio.valentinuz@deams.units.it (G. Valentinuz).

Instead, in the single-objective counterpart, one usually aims to approximate the efficient frontier by minimizing the risk for different values of the desired mean return of the portfolio. Another standard practice is to optimize a sum of the portfolio mean and variance, weighted by a risk aversion parameter. In this case, one varies the sensitivity of the investor to build up the set of optimal portfolios [14].

A third formulation maximizes the so-called Sharpe ratio, defined the ratio between the investment's excess return with respect to a risk-free rate and its standard deviation, in order to directly identify the best portfolio on the efficient frontier [46].

From a rational investor's point of view, using the Sharpe indicator in periods of market downturns is questionable because it leads to preferring riskier portfolios. A solution to this issue is to multiply, instead of dividing, this quantity by the standard deviation when the excess rate of return of the portfolio is negative (see [21]). In this way, the performance measure is coherent with agent preferences even if the risk premia are negative. A first application of this modified Sharpe ratio for a passive portfolio optimization problem with an investable universe of almost fifty assets has been proposed in [23].

In this paper, we focus on using the modified Sharpe ratio for active investments with an upper threshold on the portfolio size. We analyze the asset allocation problem from the perspective of an institutional investor who operates in large equity markets composed of hundreds or thousands of constituents and selects a restricted pool of stocks to build up a portfolio with a suitable performance with respect to the benchmark. In our portfolio design, we also consider the following standard real-world constraints. First, a budget constraint ensures that all the available capital is invested; next, bound constraints prescribe lower and upper bounds on the fraction of capital invested in each asset; finally, a turnover constraint implicitly controls the effect of the transaction costs on the rebalancing phases. The resulting mixed-integer optimization problem belongs to the general family of cardinality-constrained portfolio optimization problems which are NP-hard, and finding possible optimal solutions becomes computationally challenging [29]. For this reason, on the one hand, exact methods were proposed to supply optimal solutions, but they demand a significant amount of computation time when the problem size increases ([35] and [6]). On the other hand, heuristic approaches can identify approximate and sometimes optimal solutions within reasonable computation time even when the problem size is huge [41].

Many other real-world and engineering applications can also be modelled as highly constrained optimization problems ([16] and [37]). The lack of a universal optimizer [20] suggests the development of efficient algorithmic designs which exploit the problem-specific features. For this purpose, hyper-heuristic, memetic, and ensemble algorithms have been recently introduced (see [7] and [42]). It is worth noting that these meta-heuristics are all linked and address the same topic from different perspectives. More specifically, the first type of solvers consists of different search algorithms equipped by a coordination engine that selects and activates the various algorithms for the problem at hand. The second class combines the benefits of ad hoc local search heuristics and multi-agent systems in the algorithmic framework. Finally, an ensemble of strategies involves multiple and complementary search techniques to handle the optimization problem issues.

Within the context of memetic optimizers, swarm optimization algorithms, inspired by the self-organizing interaction among agents, have become popular in portfolio selection theory [18]. In particular, the particle swarm optimization (PSO) algorithm has been widely employed to solve real-world financial problems since its first proposal ([15], [46] and [12]) due to its effectiveness in reaching optimal solutions. However, the algorithm above does not work efficiently when the problem size is large, leading to population stagnation and premature convergence [31]. To improve PSO performance for large-scale optimization problems, several authors have designed many variants, such as competitive swarm optimizer [10], social learning particle swarm optimizer [11] and level-based learning swarm optimizer (LLSO) [44]. In particular, the latter one has shown better exploitation ability in different environments. For this reason, we propose its use to solve our portfolio optimization problem. LLSO is inspired by the teaching concept that teachers should treat students differently according to their cognitive and learning abilities. Based on that, the general idea of the LLSO is to sort the swarm individuals in ascending order with respect to their fitness and then separate them into distinct levels. The best individuals are stored at a higher level and are not updated, preserving the most valuable information conveyed in the swarm. Unlike PSO, which uses the historically best positions to update the particles, LLSO employs predominant particles in the current swarm to guide the learning of the worst particles and to enhance the swarm diversity. Thus, particles in lower levels have more individuals in the upper levels to learn from and are focused on exploring the search space; those in higher levels mainly concentrate on the exploitation task. Even though LLSO shows promising capabilities in dealing with large-scale optimization problems, it is overly sensitive to its parameters. To mitigate this influence, an adaptive variant, henceforth ALLSO, has been introduced in [38], which takes advantage of a swarm aggregation indicator to estimate the evolution state of the swarm. Two adaptive adjustment strategies are then applied to identify the best configuration setting for each generation.

Due to the fact that the swarm optimization algorithms are usually blind to the constraints, they have to be equipped with constraint-handling techniques [28] to be effective in real-world applications. A class of constraint-handling methods widely used in literature is represented by the penalty function methods, where a penalty term reduces the fitness value of the infeasible candidates. However, despite its simplicity, this method usually requires the definition of problem-dependent parameters that significantly impact algorithm performance. To overcome this issue, adaptive penalty techniques have been developed, in which the parameters are automatically set by using information gathered from the violated constraints at the current generation. We refer the reader to [2] for a more exhaustive overview of adaptive penalty techniques.

Therefore, to tackle the presented large-scale cardinality-constrained portfolio optimization problem, our memetic algorithm combines an improved ALLSO with a novel hybrid constraint-handling technique, in which we integrate a projection operator into the self-adaptive penalty scheme developed by [13]. Moreover, to further improve the algorithm's exploitation power and the solutions' quality, we introduce a novel mutation procedure, applied to the best individuals in the first level, which generalizes the one inspected in [23].

The contributions of the paper can be summarised as follows:

- we develop a memetic algorithm integrating the ALLSO with an ad hoc mutation operator and a novel hybrid constraint-handling technique to solve large-scale cardinality-constrained portfolio optimization problems;
- we use a modified version of the Sharpe ratio as the objective function in active portfolio optimization;
- we introduce a novel initialization procedure for population-based heuristics to effectively address the low degree of feasibility in our portfolio rebalancing problem;
- we validate the profitability of the investment strategy in the last five years using a pool of 1119 assets from a global equity market index.

Let us now give a more precise overview of the paper's contents. The following section describes the investment framework, focusing on the portfolio optimization problem. In Section 3, we introduce the developed solver. More precisely, we first explain the adaptive LLSO and then detail the proposed methods, namely the novel mutation operator and the hybrid constraint-handling technique. In the last part of the section, we summarise the entire procedure. In Section 4, we show the experimental results; in the last section, we depict the conclusions and future perspectives.

2. Portfolio design

2.1. Investment framework

We consider a frictionless market in which no short selling is allowed, and all investors act as price takers. Assuming that n assets represent the investable universe, a portfolio is identified with the vector of assets weights $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$, where $x_i \in \mathbb{R}$ denotes the proportion of capital invested in asset i, with $i = 1, \dots, n$. Let R_i be the random variable which stands for the rate of return of asset i, with expected value μ_i. Hence, the random variable $R_p(\boldsymbol{x}) = \sum_{i=1}^n R_i x_i$ indicates the rate of return of portfolio $\boldsymbol{x}$. The expected rate of return of portfolio $\boldsymbol{x}$ is then defined as

$$\mu_p(\boldsymbol{x}) = \sum_{i=1}^n x_i \mu_i \tag{2.1}$$

and its standard deviation, also called volatility, is given by

$$\sigma_p(\boldsymbol{x}) = \sqrt{\sum_{i=1}^n \sum_{j=1}^n c_{ij} x_i x_j} \tag{2.2}$$

where $(C)_{ij} = c_{ij}$ is the covariance between stocks i and j, with $i, j = 1, \dots, n$.

Since investors perceive large deviations from the portfolio mean value as damaging, (2.2) represents the so-called portfolio risk.

In such a setting, the portfolio choice is made only with respect to the expected portfolio rate of return and the portfolio risk, as stated in the following definition.

Definition 2.1. Given two portfolios $\boldsymbol{x}, \boldsymbol{y}$, we say that $\boldsymbol{x}$ is preferred to $\boldsymbol{y}$ if and only if $\mu_p(\boldsymbol{x}) \geq \mu_p(\boldsymbol{y})$ and $\sigma_p(\boldsymbol{x}) \leq \sigma_p(\boldsymbol{y})$, with at least one strict inequality.

In other words, an investor prefers one portfolio to another if it has a higher expected rate of return and lower risk. This decision making approach is known as mean-variance analysis.

2.2. Objective function

According to the mean-variance analysis, we can use the so-called Sharpe ratio to identify the best investment among efficient portfolios. As stated before, it is defined as the ratio between the excess return of an investment with respect to a risk-free rate, r_f, and its standard deviation

$$SR(\boldsymbol{x}) = \frac{\mu_p(\boldsymbol{x}) - r_f}{\sigma_p(\boldsymbol{x})}. \tag{2.3}$$

This performance measure evaluates the compensation earned by the investor per unit of both systematic and idiosyncratic risks [9]. Thus, higher values of SR indicate more promising portfolios.

From a theoretical point of view, this choice is justified by the fact that several widely used performance measures are increasing functions of the Sharpe ratio ([33] and [34]). Moreover, when the numerator in (2.3) is positive, this indicator is coherent with the risk-return profile of a rational investor. From a practical point of view, it can be easily calculated and its interpretation is simpler than most of recently proposed complex performance measures [1].

However, as pointed out in [21], the reliability of this performance measure decreases when the excess rate of return is negative. In that case, one would prefer higher-risk portfolios using the Sharpe ratio. To overcome this issue, in our portfolio selection problem we adopt as objective function the following modification of (2.3), the so-called modified Sharpe ratio

$$MSR(\boldsymbol{x}) = \frac{\mu_p(\boldsymbol{x}) - r_f}{\sigma_p(\boldsymbol{x})^{\text{sign}(\mu_p(\boldsymbol{x}) - r_f)}} \tag{2.4}$$

where $\text{sign}(z)$ is the sign function of $z \in \mathbb{R}$. Observe that if the portfolio excess return is non-negative, the modified Sharpe ratio is equal to the Sharpe ratio. Otherwise, it multiplies the portfolio excess return by the standard deviation. In this manner, even in adverse conditions, portfolios with lower risk and higher excess return will be preferred.

2.3. Constraints

In our portfolio model, we consider the following constraints.

- *Budget*. All the available capital needs to be invested. In terms of portfolio weights, this translates to

$$\sum_{i=1}^{n} x_i = 1. \tag{2.5}$$

- *Cardinality*. We assume that the portfolio includes up to k assets, where $k \leq n$. To model the inclusion or the exclusion of the i-th asset in the portfolio, a binary variable δ_i is introduced as

$$\delta_i = \begin{cases} 0, \text{ if asset } i \text{ is excluded} \\ 1, \text{ if asset } i \text{ is included} \end{cases} \tag{2.6}$$

for $i = 1, \ldots, n$. The resulting vector of selected assets is $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_n) \in \{0,1\}^n$, and the cardinality constraint can be written as

$$\sum_{i=1}^{n} \delta_i \leq k. \tag{2.7}$$

- *Box*. A balanced portfolio should avoid extreme positions and foster diversification. Hence, we impose a maximum and a minimum limit for portfolio weights, that is

$$\delta_i l_i \leq x_i \leq \delta_i u_i, \quad i = 1, \ldots, n \tag{2.8}$$

where l_i and u_i are the lower and the upper bounds for the weight of the i-th asset, respectively, with $0 < l_i < u_i \leq 1$ to exclude short sales.

- *Turnover*. To control the effect of the transaction costs in the portfolio rebalancing phases, we consider a portfolio turnover constraint. Let $\boldsymbol{x}_0$ be a vector containing the current portfolio positions [36]. Then, the portfolio turnover constraint is

$$\sum_{i=1}^{n} |x_i - x_{0,i}| \leq TR \tag{2.9}$$

where TR denotes the maximum turnover rate, which lies between 0 and 1. Note that if $TR = 0$ rebalancing is not allowed, and more trades are allowed when TR increases.

The pairs $(\boldsymbol{\delta}, \boldsymbol{x}) \in \{0,1\}^n \times \mathbb{R}^n$ that satisfy (2.5), (2.7), (2.8) and (2.9) form the feasible set $\mathcal{F}$. Then, our portfolio optimization problem can be written as

$$\begin{aligned} \max_{\boldsymbol{\delta}, \boldsymbol{x}} \quad & MSR(\boldsymbol{x}) \\ \text{s.t.} \quad & (\boldsymbol{\delta}, \boldsymbol{x}) \in \mathcal{F}. \end{aligned} \tag{2.10}$$

Remark 2.2. To simplify the following treatment, we reformulate our maximization problem into the equivalent minimization problem

$$\begin{aligned} \min_{\boldsymbol{\delta}, \boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & (\boldsymbol{\delta}, \boldsymbol{x}) \in \mathcal{F} \end{aligned} \tag{2.11}$$

where $f(\boldsymbol{x}) = -MSR(\boldsymbol{x})$.

3. Optimization algorithm

3.1. Adaptive level-based learning swarm optimizer

The algorithm evolves a swarm of NP candidate solutions using the so-called level-based population structure [44], according to which the evolution process is defined as follows.

1. At each iteration g, the individuals in the swarm are first sorted ascending based on their fitness and grouped into NL_g levels, each one containing $LP_g = \lfloor NP/NL_g \rfloor$ particles. In the last level, there are $\lfloor NP/NL_g \rfloor + NP\%NL_g$ particles.[1] Better individuals belong to higher levels, and a higher level corresponds to a smaller level index. Thus, L_1 represents the best level, while L_{NL_g} is the worst one.

2. To preserve the most valuable information conveyed in the current swarm, individuals belonging to L_1 are not updated and enter directly in the next generation. The p-th particle in level L_l, denoted by $\boldsymbol{x}^{l,p}(g)$, where $l = 3, \dots, NL_g$ and $p = 1, \dots, LP_g$, is allowed to learn from two particles $\boldsymbol{x}^{l_1,p_1}(g)$, $\boldsymbol{x}^{l_2,p_2}(g)$ randomly extracted from two different higher levels L_{l_1} and L_{l_2} with $l_1 < l_2$, and p_1 and p_2 are randomly chosen from $\{1, \dots, LP_g\}$. For $l = 2$, we sample two particles from L_1 in such a way that $\boldsymbol{x}^{l_1,p_1}(g)$ is better than $\boldsymbol{x}^{l_1,p_2}(g)$ in terms of fitness function. Thus, the update rule for particle $\boldsymbol{x}^{l,p}(g)$ is given component-wise by

$$v_i^{l,p}(g+1) = r_1 v_i^{l,p}(g) + r_2 \left(x_i^{l_1,p_1}(g) - x_i^{l,p}(g) \right) + \phi_g r_3 \left(x_i^{l_2,p_2}(g) - x_i^{l,p}(g) \right) \tag{3.1}$$

$$x_i^{l,p}(g+1) = x_i^{l,p}(g) + v_i^{l,p}(g+1) \tag{3.2}$$

for $i = 1, \dots, n$, where $v_i^{l,p}(g)$ denotes the i-th component of the velocity of particle p in level L_l at generation g, and r_1, r_2, r_3 are real numbers randomly generated within $[0,1]$. The parameter $\phi_g \in [0,1]$ controls the influence of the less performing exemplar $\boldsymbol{x}^{l_2,p_2}(g)$ on $\boldsymbol{v}^{l,p}(g)$.

Based on [38], both the parameters involved in the learning process at generation g, namely NL_g and ϕ_g, are adaptively adjusted based on the evolution state of the swarm by an aggregation indicator, which is defined as

$$s(g) = \frac{\overline{f}_g - f\left(\boldsymbol{x}_{gbest}(g) \right)}{f\left(\boldsymbol{x}_{gbest}(g) \right) + \xi} \tag{3.3}$$

where $\overline{f}_g$ is the average fitness of the population at generation g, $f\left(\boldsymbol{x}_{gbest}(g) \right)$ denotes the historically global best fitness up to iteration g, and ξ is a small positive value to avoid zero denominators.

Remark 3.1. When $s(g)$ is high, particles are far from the current global best solution. Thus, the swarm is in an exploration phase. On the contrary, when $s(g)$ is low, particles are close to the global best solution $\boldsymbol{x}_{gbest}(g)$ and the swarm is in an exploitation phase.

To guarantee a control on the number of levels, NL_g takes values in the set $\{NL_{min}, \dots, NL_{max}\}$, where $NL_{min}, NL_{max} \in \mathbb{N}$ are predefined lower and upper bounds. Moreover, to balance the level selection diversity and the exemplar diversity, NL_g can be modified only when the relative improvement of the global fitness between generation g and generation $g-1$, given by

$$t(g) = \frac{f\left(\boldsymbol{x}_{gbest}(g-1) \right) - f\left(\boldsymbol{x}_{gbest}(g) \right)}{f\left(\boldsymbol{x}_{gbest}(g) \right) + \xi}, \tag{3.4}$$

slows down or stops, which corresponds to the cases $t(g) < t(g-1)$ or $t(g) = 0$ respectively. The update of NL_g then follows the rule

$$NL_g = \begin{cases} 2 \cdot NL_{g-1} & \text{if} \quad s(g) < \bar{\delta} \\ \frac{1}{2} \cdot NL_{g-1} & \text{if} \quad s(g) \geq \bar{\delta} \end{cases} \tag{3.5}$$

where $\bar{\delta}$ is a threshold in terms of the aggregation indicator to control the adjustment of NL_g.

When NL_g is out of the range, it is adjusted as follows

$$NL_g = \begin{cases} NL_{rand} & \text{if} \quad r < px \\ NL_{max} & \text{if} \quad r \geq px \text{ and } NL_g > NL_{max} \\ NL_{min} & \text{if} \quad r \geq px \text{ and } NL_g < NL_{min} \end{cases} \tag{3.6}$$

where NL_{rand} is uniformly sampled from $\{NL_{min}, \dots, NL_{max}\}$, r is a real number randomly generated within $[0,1]$, and px is a fixed probability employed to reset NL_g.

The update for ϕ_g is designed in the following way

$$\phi_g = 0.35 + 0.1 \cdot \frac{1}{1 + 10 \cdot s(g)} \tag{3.7}$$

where $s(g)$ is the value of aggregation indicator given in (3.3).

A preliminary numerical analysis reveals that a clamping procedure, limiting the magnitude of the velocity $\boldsymbol{v}^{l,p}(g)$, provides a better exploration of the search space (in this regard, see also [30]). This function can be written component-wise as

$$v_i^{l,p}(g) = \min\{\max\{v_i^{l,p}(g), v_i^{min}\}, v_i^{max}\} \tag{3.8}$$

---

[1] We denote by $\lfloor x \rfloor$ the floor of x and by $x\%y$ the rest of the division of x by y.

where v_i^{min} and v_i^{max} are the minimum and the maximum velocity allowed for component i, with $i = 1, \ldots, n$. In the experiments, recalling equation (2.8), we set $v_i^{max} = u_i$ and $v_i^{min} = -v_i^{max}$.

3.2. Mutation operator

Instead of directly moving the individuals of the first level to the next generation, we propose to mutate them using an operator that combines two perturbation strategies properly developed for our portfolio optimization problem.

More specifically, one technique is inspired by the swap operator proposed in [25] and works as follows. First, we fix the maximum allowed number of non-null positions that could become zero, namely k_{max}^{swap}. Then, for each particle $\boldsymbol{x}^{1,p}(g)$ in level L_1 subject to swapping, we randomly sample from $\left\{1, \ldots, k_{max}^{swap}\right\}$ the number k^{swap} of non-null positions that will be set to zero. At this point, for $j = 1, \ldots, k^{swap}$, let a_j and b_j be two randomly chosen positions in $\boldsymbol{x}^{1,p}(g)$, such that $x_{a_j}^{1,p}(g) = 0$ and $l_{b_j} \leq x_{b_j}^{1,p}(g) \leq u_{b_j}$. Thus, the modified individual, $\hat{\boldsymbol{x}}^{1,p}(g)$, is defined component-wise as

$$\hat{x}_i^{1,p}(g) = \begin{cases} x_i^{1,p}(g), & \text{if } i \neq a_j \text{ and } i \neq b_j \\ l_{a_j} + \dfrac{x_{b_j}^{1,p}(g) - l_{b_j}}{u_{b_j} - l_{b_j}}(u_{a_j} - l_{a_j}), & \text{if } i = a_j \\ 0, & \text{if } i = b_j. \end{cases} \tag{3.9}$$

In this paper, based on the preliminary experiments, $k_{max}^{swap} = \lfloor 0.05 \cdot k \rfloor$, where k represents the maximum number of assets included in the portfolio.

Remark 3.2. This generalisation, allowing multiple swaps at the same time, improves the search capabilities of the original swap operator.

The other perturbation scheme focuses solely on the non-null components. For each $\boldsymbol{x}^{1,p}(g)$ to be mutated, let $I_+^{1,p}(g) = \{i : x_i^{1,p}(g) > 0\}$ then, for all $i \in I_+^{1,p}(g)$, we define the interval

$$W_i^{1,p}(g) = \left[x_i^{1,p}(g) - \Delta_i(g), x_i^{1,p}(g) + \Delta_i(g)\right] \tag{3.10}$$

where $\Delta(g) = \left(1 - \frac{g}{g_{max}+1}\right)(\boldsymbol{u} - \boldsymbol{l})$, with g_{max} be the maximum allowed number of iterations. The mutated component $\hat{x}_i^{1,p}(g)$ is randomly generated from the interval $W_i^{1,p}(g) \cap [l_i, u_i]$. For $i \notin I_+^{1,p}(g)$, we set $\hat{x}_i^{1,p}(g) = 0$. By narrowing the range over time, this procedure increases the exploration around the particles in L_1.

Remark 3.3. By construction, the solutions modified by both the perturbation operators have at most k non-null positions and satisfy the box constraints.

For each particle in L_1, the probability of applying the generalised swap operator decreases as the iteration counter increases according to the following rule

$$p_{swap}(g) = \frac{1}{1 + \exp(-0.005 \cdot g)}. \tag{3.11}$$

In the initial stages, the proposed mutation favours the global search, using the generalised swap operator to identify the most promising subset of non-null decision variables. With the progress of the generations, the role of the refinement operator increases and, in the late stages, the algorithm focuses primarily on the local search.

The pseudo-code of the developed mutation procedure is reported in Algorithm 1.

3.3. Solution coding and hybrid constraint-handling procedure

Let us introduce some notation. Let C_i denote a closed convex subset of $\mathbb{R}_+$, with $i = 1, \ldots, n$, and $K = \{i_1, \ldots, i_k\}$ be any subset of indices of $I = \{1, \ldots, n\}$ with cardinality $k \in \mathbb{N}$, so that $I \setminus K$ is its complement in I. For all $\boldsymbol{x} \in \mathbb{R}^n$, let $\boldsymbol{x}_K$ be defined component-wise as

$$\boldsymbol{x}_{K,i} = \begin{cases} x_i, & \text{if } i \in K \\ 0, & \text{if } i \in I \setminus K \end{cases} \tag{3.12}$$

and let $\pi_K : \mathbb{R}^n \to \mathbb{R}^k$ be the projection such that $\pi_K(\boldsymbol{x}) = \left(x_{i_1}, \ldots, x_{i_k}\right)$.

We start by presenting the following proposition (in this regard, see also [45]).

Proposition 3.4. Let $\boldsymbol{y} \in \mathbb{R}^n$, with $n \geq 2$. Then, the optimal K for the problem

$$\min_{\boldsymbol{x}_K : x_i \in C_i} \frac{1}{2} \|\boldsymbol{x}_K - \boldsymbol{y}\|^2 \tag{3.13}$$

Algorithm 1: Mutation procedure.

Input : $x^{1,p}(g)$, l, u, k_{max}^{swap}, g, $\Delta(g)$
Output: $\hat{x}^{1,p}(g)$

1 Set $\hat{x}^{1,p}(g) = x^{1,p}(g)$

2 Set $I^0 = \left\{ i : x_i^{1,p}(g) = 0 \right\}$

3 Set $I^+ = \left\{ i : l_i \leq x_i^{1,p}(g) \leq u_i \right\}$

4 Calculate $p_{swap}(g)$ according to (3.11)

5 **if** $rand() \leq p_{swap}(g)$ **then**

6      $k^{swap} \to \left\{ 1, \ldots, k_{max}^{swap} \right\}$

7   **for** $j = 1$ *to* k^{swap} **do**

8          $a_j \to I^0$

9          $b_j \to I^+$

10          $\hat{x}_{a_j}^{1,p}(g) = l_{a_j} + \dfrac{x_{b_j}^{1,p}(g) - l_{b_j}}{u_{b_j} - l_{b_j}} (u_{a_j} - l_{a_j})$

11          $\hat{x}_{b_j}^{1,p}(g) = 0$

12   **end**

13 **else**

14   **for** i *in* I^+ **do**

15          $lb = \max \left(x_i^{1,p}(g) - \Delta_i(g), l_i \right)$

16          $ub = \min \left(x_i^{1,p}(g) + \Delta_i(g), u_i \right)$

17          $\hat{x}_i^{1,p}(g) \to [lb, ub]$

18   **end**

19 **end**

---

is the set K^* of indices corresponding to the k largest components of y.

The proof of this result is reported in Appendix A.

In other words, the proposition states that x_{K^*} is the vector with at most k non-null components which has minimum Euclidean distance from y among all x_K, with $K \subset I$ of cardinality k.

Thanks to this projection, which implicitly enforces cardinality fulfilment, we can remove the vector of binary variables δ from the coding scheme of the solutions and we reformulate the portfolio optimization problem (2.11) only in terms of x_{K^*}. To this end, we introduce the set

$$B = \left\{ x \in \mathbb{R}^n : x_i = 0 \text{ or } x_i \in [l_i, u_i] \text{ for } i \in K^*, x_i = 0 \text{ for } i \in I \setminus K^*, \sum_{i=1}^n x_i = 1 \right\}, \tag{3.14}$$

that is the set of the points satisfying all the constraints apart from the turnover condition. Further, let $\psi(x)$ represent the value of the turnover function at x, which is given by

$$\psi(x) = \sum_{i=1}^n |x_i - x_{0,i}| - TR. \tag{3.15}$$

Then, the constrained optimization problem can be rewritten as

$$\begin{aligned} \min_{x \in B} \quad & f(x) \\ \text{s.t.} \quad & \psi(x) \leq 0. \end{aligned} \tag{3.16}$$

The following proposition, whose proof is given in Appendix A, establishes the equivalence between problem (3.16) and the mixed-integer optimization problem (2.11).

Proposition 3.5. *We assume that (δ^*, x^*) is a global solution to problem (2.11), then $x_{K^*}^*$ is a global solution to problem (3.16). Conversely, if x^* is a global solution to (3.16), then (δ^*, x^*) is a global solution to (2.11), with*

$$\delta_i^* = \begin{cases} 1 & \text{if } i \in K^* \\ 0 & \text{otherwise}. \end{cases}$$

As previously observed, the standard ALLSO algorithm can only deal with unconstrained problems; thus, we propose to incorporate a hybrid constraint-handling technique in order to solve problem (3.16).

The building block of our procedure is based on the following lemma.

Lemma 3.6. *Let $l = (l_1, \ldots, l_n)$ and $u = (u_1, \ldots, u_n)$ be such that $l_i \leq u_i$ for $i = 1, \ldots, n$. Let $y \in \mathbb{R}^n$ and define $[l, u] = \left\{ x \in \mathbb{R}^n : l_i \leq x_i \leq u_i \right\}$. Then, the orthogonal projection of y onto $[l, u]$ is given component-wise by*

$$P_{[l,u],i}(\boldsymbol{y}) = \min\{\max\{y_i, l_i\}, u_i\} \tag{3.17}$$

with $i = 1, \ldots, n$.

The derivation of the orthogonal projection $P_{[l,u]}$ can be found in [3]. We now provide the main result concerning the projection phase. We refer the reader to Appendix A for the proof.

Proposition 3.7. Let $\boldsymbol{y} \in \mathbb{R}^n$, with $n \geq 2$, and $K^{**} = \{i \in K^* : y_i > 0\}$, with K^* being the optimal set in Proposition 3.4. Assume that $\mathcal{B}$ in (3.14) is non-empty. Then, the orthogonal projection of $\boldsymbol{y}$ onto $\mathcal{B}$ is

$$P_{\mathcal{B}}(\boldsymbol{y}) = \pi_{K^{**}}^{-1}\left(P_{[\pi_{K^{**}}(l), \pi_{K^{**}}(u)]}(\pi_{K^{**}}(\boldsymbol{y} - \eta^*\boldsymbol{1}))\right) \tag{3.18}$$

where $\pi_K^{-1}(\boldsymbol{z})$ is the pre-image of $\boldsymbol{z} \in \mathbb{R}^{|K^{**}|}$ under π_K and $\eta^* \in \mathbb{R}$ is a solution of

$$\sum_{i=1}^{k} P_{[\pi_{K^{**}}(l), \pi_{K^{**}}(u)]}(\boldsymbol{y} - \eta\boldsymbol{1}) = 1. \tag{3.19}$$

Let $\mathcal{P}_g = \{\boldsymbol{x}^p(g) \in \mathbb{R}^n : p = 1, \ldots, NP\}$ be the swarm at generation g, with $g = 1, \ldots, g_{max}$. Then, the proposed ALLSO variant maps the individuals in $\mathcal{P}_g$, which are updated using (3.1) and (3.2), onto the set $\mathcal{B}$ by means of the projector defined in (3.18). The resulting mutated swarm is denoted by $\check{\mathcal{P}}_g$. Successively, we apply the self-adaptive penalty approach by [13] to handle the turnover constraint and to guarantee the global optimality of solutions. More precisely, the objective function value at each projected individual in $\check{\mathcal{P}}_g$, namely $f(\check{\boldsymbol{x}}^p)$, is normalized according to the formula

$$\hat{f}(\check{\boldsymbol{x}}^p) = \frac{f(\check{\boldsymbol{x}}^p) - f^{min}}{f^{max} - f^{min}}$$

where $f^{min} = \min_{\check{\boldsymbol{x}}^p \in \check{\mathcal{P}}_g} f(\check{\boldsymbol{x}}^p)$ and $f^{max} = \max_{\check{\boldsymbol{x}}^p \in \check{\mathcal{P}}_g} f(\check{\boldsymbol{x}}^p)$. Similarly, the corresponding normalized constraint violation is given by

$$\Psi\left(\check{\boldsymbol{x}}^p\right) = \begin{cases} \dfrac{\max\{\psi(\check{\boldsymbol{x}}^p), 0\}}{\psi^{max}}, & \text{if } \psi^{max} > 0 \\ 0, & \text{otherwise} \end{cases}$$

where ψ^{max} denotes the maximum of $\psi(\check{\boldsymbol{x}}^p)$ over all the mutated solutions in $\check{\mathcal{P}}_g$ which do not satisfy the turnover constraint.

Finally, the penalty function is

$$F\left(\check{\boldsymbol{x}}^p\right) = \begin{cases} \hat{f}\left(\check{\boldsymbol{x}}^p\right) & \text{if } \psi(\check{\boldsymbol{x}}^p) \leq 0 \\ \hat{f}(\check{\boldsymbol{z}}) + R_f \Psi\left(\check{\boldsymbol{x}}^p\right) & \text{if } \psi(\check{\boldsymbol{x}}^p) > 0 \text{ and } f(\check{\boldsymbol{x}}^p) \leq f(\check{\boldsymbol{z}}) \\ \hat{f}(\check{\boldsymbol{x}}^p) + R_f \Psi\left(\check{\boldsymbol{x}}^p\right) & \text{if } \psi(\check{\boldsymbol{x}}^p) > 0 \text{ and } f(\check{\boldsymbol{x}}^p) > f(\check{\boldsymbol{z}}), \end{cases} \tag{3.20}$$

where R_f represents the feasibility ratio for $\check{\mathcal{P}}_g$, that is the percentage of individuals in $\check{\mathcal{P}}_g$ satisfying the turnover constraint. In (3.20), the reference point $\check{\boldsymbol{z}}$ is a point belonging to $\check{\mathcal{P}}_g$ that satisfies the turnover constraint and has the lowest objective function value found so far. As in [13], if the population has no feasible points, $f(\check{\boldsymbol{z}})$ is initially and temporarily set to f^{max}, so that $f(\check{\boldsymbol{x}}^p) \leq f(\check{\boldsymbol{z}})$ for all $\check{\boldsymbol{x}}^p \in \check{\mathcal{P}}_g$ and $\hat{f}(\check{\boldsymbol{z}}) = 1$. The value of $f(\check{\boldsymbol{z}})$ is updated only when the first feasible point is encountered.

We conclude this subsection by stating the following theorem, whose proof is omitted since it is similar to the one presented in [13].

Proposition 3.8. The problem

$$\min_{\boldsymbol{x} \in \mathcal{B}} F(\boldsymbol{x})$$

with F as in (3.20), is equivalent to the problem (3.16).

3.4. Initialisation strategy and complete algorithm

Following the financial literature [5], we consider a portfolio optimization problem of high dimensions if it involves more than 400 securities. For this kind of problems, the common strategies of seeking a search space coverage by initializing the particles uniformly throughout the space as well as by increasing the size of the swarm are inefficient, because the search space grows exponentially with the dimension [39]. Moreover, the presence of highly constrained feasible regions in the search space exacerbates even more the initialization issue [17].

To effectively address the low degree of feasibility in our portfolio rebalancing problem due to the complexity of the turnover constraint, we propose a direct initialization of the candidate solutions in a neighbourhood of $\boldsymbol{x}_0$.

Let d_i^{min} and d_i^{max} be the minimum and the maximum allowed weight changes for $x_{0,i}$ respectively, with $i = 1, \ldots, n$. Let D^p denote the total portfolio weight allowed to be re-allocated in x_0 for defining the p-th candidate solution $x^p(0)$, with $p = 1, \ldots, NP$. Then, for each p,

1. we randomly select D^p within $\left[0, {}^{TR}/2\right]$;
2. we select a subset J^- of k' assets from the k assets with positive weight in x_0, so that

$$x_j^p(0) = x_{0,j} - d_j, \text{ for } j \in J^-$$

where d_j is randomly sampled in $\left[d_j^{min}, d_j^{max}\right]$ in such a way that $\sum_{j \in J^-} d_j = D^p$, and $x_j^p(0) = 0$ or $l_j \le x_j^p(0) \le u_j$;

3. we select a subset J^+ of k'' assets from the $n - k$ assets with zero weight in x_0, with $k'' \le k'$, so that

$$x_j^p(0) = x_{0,j} + d_j, \text{ for } j \in J^+$$

where d_j is randomly sampled in $\left[d^{min}, d^{max}\right]$ in such a way that $\sum_{j \in J^+} d_j = D^p$, and $l_j \le x_j \le u_j$;

4. for $j \in I \setminus \left(J^- \cup J^+\right)$, we set $x_j^p(0) = x_{0,j}$.

The portfolios assembled using this scheme satisfy cardinality, box and turnover constraints. In this way, the initialization strategy encourages the swarm to focus on exploitation rather than exploration, thereby allowing it to identify promising solutions, even in problems with high dimension and small feasible regions.

Regarding the initial velocities, we set them all equal to the zero vector, that is $v^p(0) = \mathbf{0}$, for $p = 1, \ldots, NP$.

The pseudocode of the proposed LLSO variant with adaptive parameters update, mutation of the particles in the first level and hybrid constraint-handling technique, shortly ALLSO-MUT-H, is reported in Algorithm 2. It can be noticed that, setting $x_0 = \mathbf{0}$ and $TR = 1$, ALLSO-MUT-H can also tackle portfolio optimization problems with no rebalancing. In this case, only the orthogonal projector is needed to move the unfeasible solutions to the feasible region.

A list of the main symbols used in this paper is given in Table 8.

Algorithm 2: ALLSO-MUT-H.

Input : $x_0, l, u, k, TR, NL_{min}, NL_{max}, NP, \bar{\delta}, px, \xi$
Output: x_{gbest}

1   Set $g = 0$ and $NL_g = 20$
2   Initialize the swarm $\mathcal{P}_g = \{x^p(g) : p = 1, \ldots, NP\}$ and the velocities $v^p(g)$
3   **for** $i = 1$ *to* NP **do**
4   Project $x^p(g)$ onto $\mathcal{B}$ using (3.18)
5   Calculate the turnover violation using (3.15)
6   **end**
7   Calculate the penalty F for particles in $\mathcal{P}_g$
8   Sort $\mathcal{P}_g$ by F value and divide it in NL_g levels
9   Set $x_{gbest}(g) = x^1(g)$
10   **while** $g < g_{max}$ **do**
11      $g = g + 1$
12   Set $\mathcal{P}^{L1} = \{x^p(g) : p \in L_1\}$
13   **for** $p = 1$ *to* LP_g **do**
14   Use Algorithm 1 to generate the mutated particle $\hat{x}^{1,p}(g)$ from $x^{1,p}(g)$
15   Project $\hat{x}^{1,p}(g)$ onto $\mathcal{B}$ using (3.18)
16   Calculate the turnover violation using (3.15)
17   **end**
18   Set $\mathcal{P}_{mut}^{L1} = \{\hat{x}^p(g) : p \in L_1\}$
19   Calculate the penalty F for $\mathcal{P}^{L1} \cup \mathcal{P}_{mut}^{L1}$ and update $\mathcal{P}^{L1}$ based on $\mathcal{P}_{mut}^{L1}$ using F
20   Sort $\mathcal{P}^{L1}$ by the F value
21   Calculate the swarm aggregation indicator using (3.3) and update ϕ_g using (3.7)
22   **for** $p = LP_{g+1}$ *to* NP **do**
23   Update $v^p(g)$ using (3.1) and clamp it using (3.8)
24   Update $x^p(g)$ using (3.2)
25   Project $x^p(g)$ onto $\mathcal{B}$
26   Calculate the turnover violation using (3.15)
27   **end**
28   Set $\breve{\mathcal{P}}_g$ be the set of updated particles
29   Calculate ϕ for $\mathcal{P}_g \cup \breve{\mathcal{P}}_g$ and update $\mathcal{P}_g$ based on $\breve{\mathcal{P}}_g$ using F
30   Sort $\mathcal{P}_g$ by F value
31   Calculate F for the set $\{x_{gbest}(g), x^1(g)\}$ and update $x_{gbest}(g)$
32   Calculate the relative improvement $t(g)$ of $x_{gbest}(g)$ using (3.4)
33   **if** $t(g) < t(g-1)$ *or* $t(g) = 0$ **then**
34   Update NL_g using (3.5) and (3.6)
35   **end**

Table 1
Parameter settings of the algorithms used in the comparisons.

Algorithm	Parameter settings	Reference
DLLSO	$S = \{4, 6, 8, 10, 20, 50\}$, $NP = 500$, $\phi = 0.4$	[44]
ALLSO	$NL_{min} = 2$, $NL_{max} = 50$, $NP = 500$, $\bar{\delta} = 0.01$,	
	$px = 0.01$, $\xi = 10^{-6}$	[38]
RLLPSO	$S = \{4, 6, 8, 10, 20, 50\}$, $NP = 500$, $\phi = 0.4$,	
	$\alpha = 0.4$, $\gamma = 0.8$, $\varepsilon = 0.9$, $\xi = 10^{-6}$	[40]
PSO	$\omega_{min} = 0.4$, $\omega_{max} = 0.9$, $c_{1,min} = c_{2,min} = 0.5$,	
	$c_{1,max} = c_{2,max} = 2.5$,	[32], [12]
FA	$\alpha = 0.5$, $\beta_{min} = 0.2$, $\gamma = 1$	[43]

Table 2
Data sets from [8] with the corresponding number of weeks, the number of market constituents (n) used in the estimation of parameters, and the cardinality thresholds.

Data set name	Weekly prices	Assets (n)	Cardinality threshold (k)		
			$\lfloor 30\% \cdot n \rfloor$	$\lfloor 15\% \cdot n \rfloor$	$\lfloor 5\% \cdot n \rfloor$
S&P 500	290	457	137	68	22
Russell 2000	290	1318	395	197	65
Russell 3000	290	2151	645	322	107

4. Experimental analysis

This section is divided into two parts. On the one hand, we point out the strengths and weaknesses of using the proposed algorithm to tackle large-scale cardinality-constrained portfolio optimization problems. On the other hand, we assess the profitability of the investment strategy in a real-world case study by varying the size of portfolios.

As a first task, we study the impact of the developed mutation operator on the LLSO-type algorithms, all equipped with our hybrid constraint-handling technique. For this purpose, we compare the dynamic LLSO (DLLSO) [44], the adaptive LLSO (ALLSO) [38], and the reinforcement learning level-based particle swarm optimization (RLLPSO) algorithm [40]. These are the most recent solvers proposed in the literature, that employ the level-based learning paradigm into the PSO framework. Their superiority in solution accuracy with respect to several state-of-the-art algorithms has already been proved theoretically and empirically for unconstrained optimization problems. Hence, we inspect the best variant of the LLSO-type algorithm for our constrained optimization problems. To highlight the benefits of our hybrid self-adaptive penalty approach, we compare the ALLSO-MUT-H with an ALLSO equipped by our mutation operator and the exact ℓ_1-penalty proposed in [12]. At the same time, we consider two other state-of-the-art swarm optimizers, namely PSO and FA [43], both endowed with the adaptive penalty by [12]. It is worth noting that the self-adaptive method [13] and the adaptive penalty [12] guarantee the convergence to optimal solutions.

In the numerical experiments, we follow the suggestions in the reference papers to set up the parameters of each algorithm, and the complete list of parameter values is reported in Table 1.

4.1. Algorithmic comparisons

For the algorithmic comparisons, we use three data sets from the OR-Library [8], namely S&P 500, Russell 2000, and Russell 3000, which represent large capital market indices. Table 2 summarizes the data sets employed in the experimental analysis. The second and third columns report the number of assets involved and the number of observations for each data set, respectively. The last columns show the cardinality thresholds used in the experiments. In particular, we explore the cases where k is 30%, 15%, and 5% of the size n of the corresponding data set. Since the results are similar, in the following analysis we describe only the outcomes for the cardinality threshold equal to 30% of the size of the corresponding data set. The cases regarding the last two columns of Table 2 are addressed in the supplementary file. For calculating the expected rates of return, we adopt a historical approach based on all the information available, consisting of 290 weekly prices for each asset. Since our investable universes have more decision variables than observations, the estimator of the associated covariance matrices is biased. We employ the shrinkage estimator proposed in [26] to reduce this effect. One can appreciate the difference between these estimators for the three data sets in the supplementary file.

For a fair comparison of the solvers, we adopt the following assumptions.

- All the algorithms have the same initial population of 500 individuals for each test set.
- We set $d_i^{min} = 0.0005$ and $d_i^{max} = 0.0050$ in the initialization strategy.
- For each algorithm, we perform 30 independent runs with 2000 generations, which represents the terminal condition.

Table 3
Statistics regarding the best values of the objective function over 30 runs.

Data set	Statistics	DLLSO-H	DLLSO-MUT-H	ALLSO-H	ALLSO-MUT-H	RLLPSO-H	RLLPSO-MUT-H
S&P 500	mean	-0.1526	-0.1670	-0.1524	**-0.1672**	-0.1523	-0.1622
	std	0.0004	0.0015	**0.0003**	0.0014	0.0004	0.0023
	min	-0.1532	**-0.1718**	-0.1529	-0.1698	-0.1532	-0.1662
	max	-0.1517	**-0.1643**	-0.1517	-0.1638	-0.1509	-0.1546
Russell 2000	mean	-0.1920	-0.2104	-0.1925	**-0.2124**	-0.1929	-0.2049
	std	**0.0010**	0.0042	0.0011	0.0037	0.0013	0.0043
	min	-0.1932	-0.2193	-0.1946	**-0.2196**	-0.1963	-0.2146
	max	-0.1895	-0.2003	-0.1895	**-0.2039**	-0.1903	-0.1959
Russell 3000	mean	-0.2059	-0.2247	-0.2091	**-0.2301**	-0.2082	-0.2271
	std	0.0011	0.0031	0.0015	0.0025	**0.0010**	0.0043
	min	-0.2085	-0.2328	-0.2120	-0.2352	-0.2111	**-0.2358**
	max	-0.2042	-0.2192	-0.2067	**-0.2260**	-0.2065	-0.2196

Table 4
Relative change of the mutated algorithms versus non-mutated counterparts. The p-values for the paired t-tests are displayed in brackets. Note that in all cases the p-values are under the significance level $\alpha = 0.05$, indicating the rejection of the null hypothesis of equality of the means, against the alternative left-sided hypothesis.

Data set	DLLSO-MUT-H vs. DLLSO-H (%)	ALLSO-MUT-H vs. ALLSO-H (%)	RLLPSO-MUT-H vs. RLLPSO-H (%)
S&P 500	9.4432 $(2.8638 \cdot 10^{-30})$	9.7153 $(8.6384 \cdot 10^{-31})$	6.5109 $(7.5328 \cdot 10^{-20})$
Russell 2000	9.5859 $(6.2697 \cdot 10^{-21})$	10.3564 $(2.7589 \cdot 10^{-23})$	6.2501 $(3.2532 \cdot 10^{-15})$
Russell 3000	9.1309 $(1.2435 \cdot 10^{-23})$	10.0823 $(7.4039 \cdot 10^{-29})$	9.1097 $(5.6022 \cdot 10^{-20})$

Moreover, all the portfolios from a given test set employ the following parameter setting.

- The risk-free value in (2.4) is set to zero.
- The box thresholds in (2.8) are $l_i = 0.001$ and $u_i = 0.05$ for each asset.
- Regarding the turnover constraint (2.9), TR is set equal to 0.20.
- The vector of current positions x_0 in (2.9) is fixed for all the compared algorithms and in all simulations by randomly sampling once from each set of feasible portfolios.

4.1.1. Mutation effects on LLSO variants

We recall that in the following analysis k is set always equal to 30% of the size of the corresponding data set, and we refer the interested reader to the supplementary file for the other cases.

Table 3 shows four performance metrics linked to the best objective function value over 30 runs, on the three public data sets. The best results are highlighted in bold font. We note that the ALLSO-MUT-H outperforms the competitors in all the case studies, presenting the lowest mean objective function value. Further, we remark that all the inspected LLSO variants are able to find feasible solutions. Focusing on the mutation benefits, we observe from Table 4 that the mutation has a significant impact on the performance of solvers. Specifically, although mutation-based optimizers present higher volatility than their counterparts, they always show lower results in terms of minimum-maximum range of the best solutions. Fig. 1 shows the convergence and the diversity analyses of the compared solvers on the three data sets. From the first set of graphs, we note that the three mutated algorithms are able to reach significantly lower objective function values, and the ALLSO-MUT-H performs better than the others. Moreover, the algorithms without mutation show population stagnation around 100 generations, meaning that they converge to a local minimum and are not able to further explore the search space. This is confirmed by the results showed in the logarithmic scale plots of the diversity measures. We can observe that the ALLSO-MUT-H and the DLLSO-MUT-H are able to escape from the local minima, due to the oscillatory behaviour of the swarm diversity.
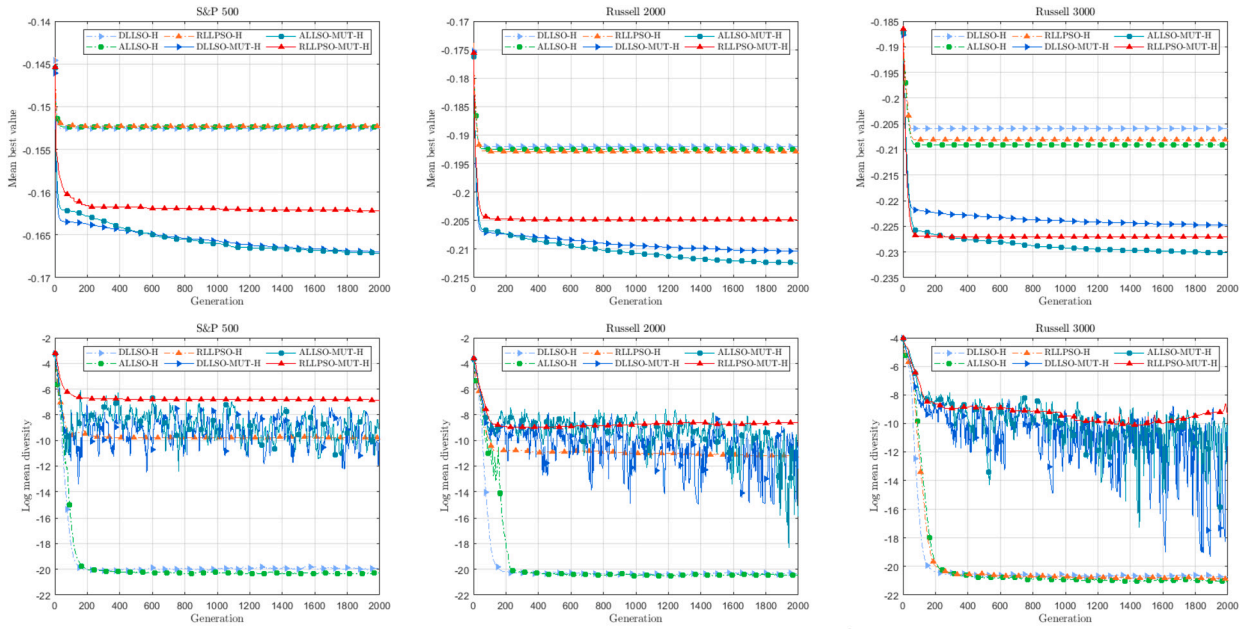
Fig. 1. Convergence and diversity analyses on the three data sets. Graphs in the first row show the behaviour of algorithms in terms of mean best value of the objective function, while in the row below are displayed the logarithmic scale plots of the diversity scores.

Table 5
Comparison with state-of-the-art swarm optimization algorithms implementing the exact ℓ_1-penalty framework proposed in [12].

Data set	Statistics	PSO-ℓ_1	FA-ℓ_1	ALLSO-MUT-ℓ_1	ALLSO-MUT-H
S&P 500	feasible sol. (%)	0	0	100	100
	mean CV	$3.1572 \cdot 10^{-12}$	$3.1572 \cdot 10^{-12}$	0	0
	mean F_{ℓ_1}	-0.1325	-0.1325	-0.1325	-0.1672
Russell 2000	feasible sol. (%)	63	100	100	100
	mean CV	$1.1661 \cdot 10^{-5}$	0	0	0
	mean F_{ℓ_1}	-0.1717	-0.1639	-0.1639	-0.2124
Russell 3000	feasible sol. (%)	83	100	0	100
	mean CV	$7.4015 \cdot 10^{-12}$	0	0.0326	0
	mean F_{ℓ_1}	-0.1862	-0.1291	-0.1487	-0.2301

4.1.2. Comparison with state-of-the-art swarm optimization algorithms

In the previous subsection, we have analysed the impact of the mutation on the capabilities of LLSO-based algorithms, finding that the ALLSO-MUT-H is the more efficient choice in terms of convergence and quality of solutions. We recall also that we adopt the same parameter setup presented above in Table 1. We exhibit the statistics of the comparison in Table 5, where are displayed the percentage of feasible solutions provided by the different solvers over the 30 runs; the mean of the constraint violation function CV for the non-feasible solutions; the average value of the penalty function F_{ℓ_1} over 30 runs. Notice that, the penalty function corresponds to the objective function when the solutions are feasible. Looking at the results, we can argue that the ALLSO-MUT-H reaches the best mean value of the penalty function in all the data sets, and it always provides feasible solutions. This insight is confirmed by the convergence analysis plots in Fig. 2, which show the benefits of our constraint-handling technique in terms of accuracy of solutions. Moreover, the diversity graphs suggest that our solver is the sole algorithm able to exhibit exploration and exploitation phases alternatively.
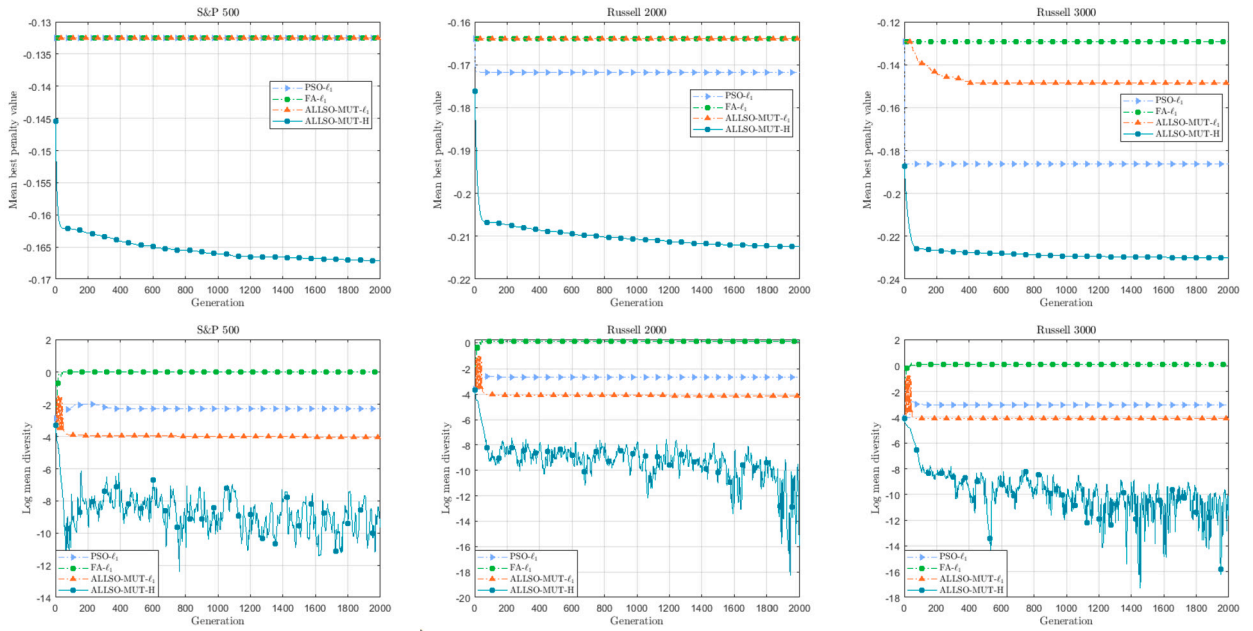
Fig. 2. Plots in the first row show the behaviour of the algorithms in terms of mean best value of the penalty function, while in the second row are presented the logarithmic scale graphs of the diversity.

4.2. Real-world application

4.2.1. Data description and investment setting

The constituents of the MSCI World index, on 31st January 2022, form our investable pool. The data set has been downloaded from DataStream and consists of monthly prices covering the period from January 2012 to January 2022 for a total of 121 months. Stocks with missing observations were disregarded, and thus the final data set includes 1119 stocks. For the performance comparisons, we introduce a value-weighted benchmark index with the same constituents and, as in the previous analysis, we set $r_f = 0$.

The portfolio design employs the following parameter setting. For the cardinality constraint (2.7), we consider $k \in \{335, 167, 55, 22\}$, corresponding to $k_\% = 30\%$, 15%, 5% and 2% of the pool size, respectively. As stated in the introductory part of Section 4.1, we recall that the box thresholds in (2.8) are $l_i = 0.001$ and $u_i = 0.05$ for each asset i, with $i = 1, \ldots, 1119$, and the turnover rate in (2.9) is set equal to 0.20, as in [23].

We use a rolling time window procedure to rebalance optimal portfolios every month, from January 2017 to January 2022, to point out the effects of the market changes on the behaviour of the investments and, as a consequence, the total number of ex-post dates is 61. We solve the corresponding problem instances by employing overlapping 60-months windows, which are updated every month by removing the oldest data and including the latest information.

In each quoted window, as already pointed out above, we adopt a historical approach to calculate expected rates of return, and to reduce the bias in the estimation of the covariance matrix C, we take advantage of the shrinkage estimator proposed in [26]. Let us denote by $\boldsymbol{x}_t$ the optimal portfolio at the ex-post month t, with $t = 1, \ldots, 61$. Due to the time dependence of the considered investment plan, we rewrite the turnover constraint (2.9) as follows

$$\sum_{i=1}^{n} |x_{t,i} - x_{t^-,i}| \leq TR. \tag{4.1}$$

In the previous equation, $\boldsymbol{x}_{t^-} = (x_{t^-,1}, \ldots, x_{t^-,n})$ represents the portfolio to be rebalanced [36], which is defined for $t = 2, \ldots, 61$ as

$$x_{t^-,i} = \frac{x_{t-1,i} R_{t-1,i}^g}{\sum_{j=1}^{n} x_{t-1,j} R_{t-1,j}^g} \tag{4.2}$$

with the denominator being the gross portfolio return at month $t - 1$.[2] At time $t = 1$, we set $\boldsymbol{x}_{t^-} = \boldsymbol{0}$ and $TR = 1$.

Let us assume a self-financing strategy with an initial wealth $W_0 = 10,000,000$ \$. Then, we explicitly evaluate the magnitude of the trading through the cost function $\lambda(\boldsymbol{x}_t, \boldsymbol{x}_{t^-})$ introduced in [4]. As reported in Table 6, we consider the transaction cost structure characterized by decreasing cost rates as the traded value increases.

---

[2] The gross return of asset i at month t is defined as $R_{i,t}^g = \frac{S_{i,t}}{S_{i,t-1}}$, where $S_{i,t}$ is the price of the i-th asset at the end of month t.

Table 6
Structure of transaction costs.

Trading segment ($)	Fixed fee ($)	Proportional cost (%)
0 – 7,999	40	0
8,000 – 49,999	0	0.5
50,000 – 99,999	0	0.4
100,000 – 199,999	0	0.25
$\geq 200,000$	400	0

4.2.2. Ex-post performance measures

The following measures (see [22] and [23]) are considered to evaluate the profitability of the investment strategies. Let $r_{p,t}^{out}$ be the ex-post portfolio rate of return realized at time t, with $t = 1, \dots, 61$. First, we consider the so-called ex-post Sharpe ratio, defined as

$$SR^{out} = \frac{\mu^{out}}{\sigma^{out}} \tag{4.3}$$

where μ^{out} and σ^{out} are the mean and the standard deviation of the ex-post portfolio rates of return, respectively.

The second measure employed in the analysis is the so-called Omega ratio, defined as the ratio between the gains over a threshold level and the losses under a threshold level. In this study, we set both thresholds equal to zero, that is

$$Omega = \frac{\sum_{t=1}^{61} r_{p,t}^{out} \mathbb{1}_{\{r_{p,t}^{out} > 0\}}}{-\sum_{t=1}^{61} r_{p,t}^{out} \mathbb{1}_{\{r_{p,t}^{out} < 0\}}} \tag{4.4}$$

where $\mathbb{1}_A$ is the indicator function on A.

The information gathered from these performance measures draws a complete picture of the ex-post portfolio return distribution. In particular, the ex-post Sharpe ratio describes the central part of the portfolio return distribution, while the Omega ratio considers the behaviour of profits and losses.

Further, to measure the profitability of the investment at time t, we compute the net wealth as

$$W_t = W_{t-1}\left(1 + r_{p,t}^{out}\right) - \lambda(\boldsymbol{x}_t, \boldsymbol{x}_{t^-}). \tag{4.5}$$

Then, we compare the profitability of the investments using the so-called compound annual growth rate, which in our case is calculated as

$$CAGR = \left(\frac{W_{61}}{W_0}\right)^{\frac{12}{61}} - 1 \tag{4.6}$$

where W_0 represents the initial wealth and W_{61} is the final wealth.

To evaluate the capacity of a strategy to avoid high losses, we introduce the drawdown measure, which can be written

$$DD_t = \min\left\{0, \frac{W_t - W_{peak}}{W_{peak}}\right\} \tag{4.7}$$

where W_{peak} is the maximum amount of wealth reached by the strategy until time t. In particular, we consider the mean and the standard deviation of the drawdown measure over time.

Finally, we propose to measure the effect of the costs on the available capital in the out-of-sample period by

$$\Lambda_\% = \frac{1}{61} \sum_{t=1}^{61} \frac{\lambda(\boldsymbol{x}_t, \boldsymbol{x}_{t^-})}{W_{t-1}} \cdot 100. \tag{4.8}$$

4.2.3. Ex-post performance analysis

In the ex-post analysis, we investigate how the performance of the proposed asset allocation model changes by varying the k parameter in (2.7).

First, we remark that, for any ex-post dates, the proposed hybrid LLSO variant identifies feasible solutions for all the portfolio sizes. The empirical results are summarised in Table 7, where the number of assets of the considered strategies is also displayed. Note that for each value of k, the proposed investments provide better performances than the value-weighted benchmark. This result implies that introducing a cardinality constraint in the portfolio model allows to choose a subset of the most profitable assets in the investable pool.

In terms of the return-risk profile, strategies with $k_\% = 30\%$, 15%, and 5% show comparable performances, while the strategy with $k_\% = 2\%$ has a lower Sharpe ratio, which is due to its large volatility. Similar conclusions can be made about the Omega ratio, which expresses the gain-loss profile of the strategies. Despite better performance with respect to Sharpe and Omega ratios, strategies

Table 7

Performance of the proposed cardinality-constrained portfolio allocation model for different cardinalities in comparison to the benchmark.

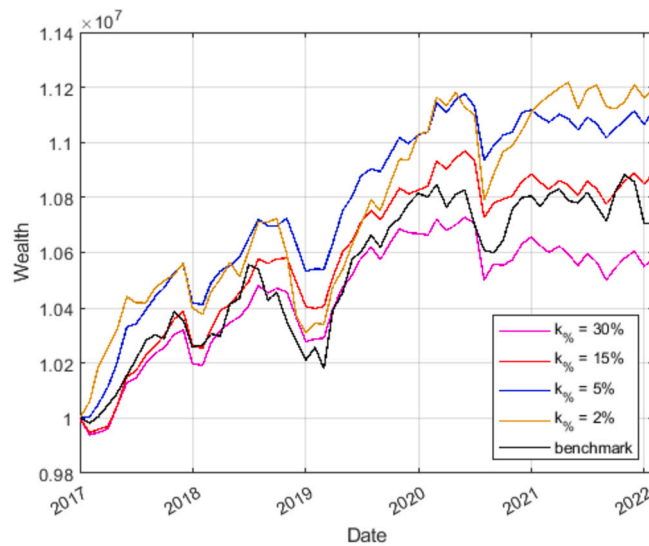| $k_{\%}$	30%	15%	5%	2%	Benchmark
num. assets	335	167	55	22	1119
Sharpe ratio	0.4917	0.4794	0.4467	0.2878	0.1731
Omega ratio	3.3269	3.2428	3.0360	2.0980	1.5388
CAGR	1.1148	1.6899	2.1118	2.2502	1.3460
std	0.0052	0.0052	0.0054	0.0077	0.0066
mean DD	-0.0066	-0.0057	-0.0050	-0.0072	-0.0070
std DD	0.0070	0.0064	0.0060	0.0106	0.0089
mean λ ($)	16,046	10,713	6,571.7	3,121.9	–
$\Lambda_{\%}$	0.1536	0.1011	0.0610	0.0290	–

Fig. 3. Ex-post evolution of net wealth of the benchmark and of the proposed cardinality-constrained portfolio allocation model with different cardinalities.

involving portfolios with a larger number of assets generate less wealth. Moreover, we observe that reducing cardinality leads to more profitable portfolio strategies.

Concerning the drawdown measures, the 5% asset allocation model is the most conservative, while the one with $k_{\%} = 2\%$ is the worst. Thus, the performance deteriorates by reducing portfolio size below a critical threshold.

As highlighted in the last two rows of Table 7 and in Fig. 3, the impact of transaction costs for strategies with small k is negligible. On the contrary, portfolios with many assets have more fluctuations in the rebalancing phases, leading to higher trading commissions with a significant impact on the wealth generated.

Summing up, we can infer that the strategy with $k_{\%} = 5\%$ shows the best balance between risk-adjusted performance measures and capability to generate net profits.

5. Conclusions and future works

In this paper, we have developed a swarm optimization algorithm for solving a large-scale cardinality-constrained portfolio optimization problem, where a modified Sharpe ratio performance measure represents the objective function. We have considered four real-world constraints: cardinality, box, budget, and turnover constraints. Due to the properties of the model inspected, we have proposed a variant of the LLSO equipped with a hybrid procedure to manage the constraints efficiently. Moreover, a novel mutation operator has been introduced to improve the accuracy of solutions. Our solver capabilities have been compared with those of two variants of the LLSO as well as other state-of-the-art swarm optimization algorithms endowed with an ℓ_1-penalty function. Numerical experiments on three publicly available large-scale data sets showed the outperformance of our hybrid procedure. From the financial point of view, we have analyzed the sensitivity of the portfolio model to the cardinality constraint with data from the last five years of the MSCI World index. We have found that portfolios of small size are more competitive with respect to the value-weighted benchmark index, also in periods of market downturns. Specifically, the losses are reduced, and the cost impact on the available capital is marginal compared to the profits.

The developed algorithm has a direct application to solving financial management problems. However, it could be easily modified for large-scale parameter optimization, such as for training feed-forward neural networks. Moreover, since the projection paradigm has already been used profitably to tackle real-world problems in the engineering field like sensor networks, radiation therapy treatment planning, computerized tomography, magnetic resonance imaging, and optics to name a few, in our future research we plan to extend our portfolio optimization model by including other constraints. On the one hand, we will consider the so-called risk-budgeting constraints to control the portfolio risk exposition explicitly. On the other hand, based on the European Green Deal and the ESG Disclosure requirements for funds and investments, we will add sustainable-policy constraints to guarantee a minimum level of ESG rating to the investment. The inclusion of these new constraints in the optimization model requires an adequate adaptation of the hybrid-constraint handling technique, by considering novel projection schemes. Another possible research direction to fill the gap in handling the tail-risk is represented by the introduction of alternative performance measures in the optimization framework, like the conditional value-at-risk and its related measures.

CRediT authorship contribution statement

The authors confirm that all have contributed equally to the paper for study conception and design, data collection, analysis and interpretation of results, and draft manuscript preparation.

All authors reviewed the results and approved the final version of the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgements

Appendix A. Proofs of the main results

Proof of Proposition 3.4. We rewrite problem (3.13) in the following way

$$\min_{\boldsymbol{x}_K \, : \, x_i \in C_i} \left\{ \frac{1}{2} \sum_{j \in K} (x_j - y_j)^2 \right\} + \frac{1}{2} \sum_{j \in I \setminus K} y_j^2,$$

that is equivalent to

$$\min_{\boldsymbol{x}_K \, : \, x_i \in C_i} \left\{ \frac{1}{2} \sum_{j \in K} (x_j - y_j)^2 \right\} - \frac{1}{2} \sum_{j \in K} y_j^2 + \frac{1}{2} \sum_{j \in I} y_j^2. \tag{A.1}$$

We note that the last term in (A.1) does not depend on x_j, $j \in K$, so we can focus our attention on the first two terms, i.e.

$$\min_{\pi_K(\boldsymbol{x}) \, : \, x_i \in C_i} \frac{1}{2} \| \pi_K(\boldsymbol{x} - \boldsymbol{y}) \|^2 - \frac{1}{2} \| \pi_K(\boldsymbol{y}) \|^2.$$

By contradiction, we suppose that there is a K' different from K^*, where we recall that K^* is the set of indices corresponding to the k largest components of $\boldsymbol{y}$. At this point, we define

$$f(\pi_K(\boldsymbol{y})) := -\frac{1}{2} \| \pi_K(\boldsymbol{y}) \|^2 + \min_{\pi_K(\boldsymbol{x}) \, : \, x_i \in C_i} \frac{1}{2} \| \pi_K(\boldsymbol{x} - \boldsymbol{y}) \|^2$$

and

$$g(t) := f((1 - t)\pi_{K^*}(\boldsymbol{y}) + t\pi_{K'}(\boldsymbol{y})) \quad \text{with} \quad t \in [0, 1].$$

Then we have,

$$f(\pi_{K'}(\boldsymbol{y})) - f(\pi_{K^*}(\boldsymbol{y})) = g(1) - g(0) = \int_0^1 g'(t)\, \mathrm{d}t$$

and

$$g'(t) = \nabla f((1-t)\pi_{K^*}(\mathbf{y}) + t\pi_{K'}(\mathbf{y})) \cdot (-\pi_{K^*}(\mathbf{y}) + \pi_{K'}(\mathbf{y})),$$

where $\nabla f(\pi_K(\mathbf{y})) = -\pi_K(\mathbf{y}) + \pi_K(\mathbf{y}) - \pi_K(\mathbf{x}^*) = -\pi_K(\mathbf{x}^*)$ and $\pi_K(\mathbf{x}^*) = \underset{\pi_K(\mathbf{x}):\, x_i \in C_i}{\mathrm{argmin}} \; \frac{1}{2}\|\pi_K(\mathbf{x} - \mathbf{y})\|^2$. Now, since $C_i \subset \mathbb{R}_+$, we have that $\nabla f(\pi_K(\mathbf{y}))$ is non-positive in all components. Moreover, $-\pi_{K^*}(\mathbf{y}) + \pi_{K'}(\mathbf{y}) \leq 0$ due to the fact that $\pi_{K^*}(\mathbf{y})$ is the projection of $\mathbf{y}$ onto the set of its largest components. As a result, we obtain $g'(t) \geq 0$, which implies $f(\pi_{K'}(\mathbf{y})) \geq f(\pi_{K^*}(\mathbf{y}))$. This means that K^* must be the optimal choice. $\quad\square$

Proof of Proposition 3.5. The proof of the first part of the proposition follows by defining $\mathbf{x}_{K^*}^*$ such that $\mathbf{x}_{K^*}^* = \boldsymbol{\delta}^* \otimes \mathbf{x}^*$, where $\otimes$ stands for the Hadamard product.

On the contrary, if $\mathbf{x}^*$ solves (3.16), then $\mathbf{x}_{K^*}^* = \mathbf{x}^*$. By taking

$$\delta_i^* = \begin{cases} 1 & \text{if} \quad i \in K^* \\ 0 & \text{otherwise} \end{cases}$$

we deduce that $(\boldsymbol{\delta}^*, \, \mathbf{x}^*)$ solves (2.11). $\quad\square$

Proof of Proposition 3.7. The result follows from Theorem 6.27 in [3], where the hyperplane is represented by the budget constrain (2.5) and the box is $[\pi_{K^{**}}(\mathbf{l}), \pi_{K^{**}}(\mathbf{u})]$. We recall also that $\pi_{K^{**}}(\mathbf{z}) = (z_{i_1}, \ldots, z_{i_k})$ with $z_{i_j} > 0$ and $i_j \in K^{**}$.

Appendix B. An approach based on an exact ℓ_1-penalty function

In this appendix we introduce a procedure based on the exact ℓ_1-penalty function for solving cardinality-constrained portfolio optimization problems. We adapt the approach discussed in [12] to the algorithms used in the comparison analysis of Subsection 4.1 and thus we define the constraint violations as follows

$$CV_1 = \left| \sum_{i=1}^n x_i - 1 \right|$$

$$CV_2 = \max\left\{ \sum_{i=1}^n \delta_i - k, 0 \right\}$$

$$CV_3 = \sum_{i=1}^n \max\left\{ \delta_i l_i - x_i, 0 \right\}$$

$$CV_4 = \sum_{i=1}^n \max\left\{ x_i - \delta_i u_i, 0 \right\}$$

$$CV_5 = \sum_{i=1}^n \left| \delta_i(1 - \delta_i) \right|$$

$$CV_6 = \max\left\{ \sum_{i=1}^n \left| x_i - x_{0,i} \right| - TR, 0 \right\}.$$

In this manner, we introduce the exact ℓ_1-penalty function

$$F_{\ell_1}(\mathbf{x}, \boldsymbol{\delta}; \boldsymbol{\varepsilon}) = f(\mathbf{x}) + \frac{1}{\varepsilon_0}\left[\varepsilon_1 CV_1 + \varepsilon_2 CV_2 + \varepsilon_3 CV_3 + \varepsilon_4 CV_4 + \varepsilon_5 CV_5 + \varepsilon_6 CV_6 \right] \tag{B.1}$$

where $\boldsymbol{\varepsilon} = (\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_6)$, with $\varepsilon > 0$ for all i.

The initial parameters vector $\boldsymbol{\varepsilon}^0$ is set to $\boldsymbol{\varepsilon}^0 = (\varepsilon_0^0, \varepsilon_1^0, \ldots, \varepsilon_6^0) = (10^{-4}, 1, \ldots, 1) \in \mathbb{R}^7$, where ε_0^0 is chosen in order to privilege feasible solutions, and the other parameters are equally penalized for all constraint violations.

The vector $\boldsymbol{\varepsilon}$ is updated by checking the decrease of the function $f(\mathbf{x})$ and the violation of the constraints. More precisely, on the one hand, every 5 iterations the entry $\varepsilon_0(g)$ is updated according to the rule

$$\varepsilon_0(g+1) = \begin{cases} \min\{3 \cdot \varepsilon_0(g), 1\} & \text{if} \quad f(\mathbf{x}(g)) \geq f(\mathbf{x}(g-1)) \\ \max\{0.6 \cdot \varepsilon_0(g), 10^{-15}\} & \text{if} \quad f(\mathbf{x}(g)) < 0.9 \cdot f(\mathbf{x}(g-1)) \\ \varepsilon_0(g) & \text{otherwise.} \end{cases} \tag{B.2}$$

On the other hand, every 10 iterations the entries $\varepsilon_i(g)$, $i = 1, \ldots 6$, are updated following the scheme

$$\varepsilon_i(g+1) = \begin{cases} \min\{2 \cdot \varepsilon_i(g), 10^4\} & \text{if} \quad CV_i(g) > 0.95 \cdot CV_i(g-1) \\ \max\{0.5 \cdot \varepsilon_i^g, 10^{-4}\} & \text{if} \quad CV_i(g) < 0.9 \cdot CV_i(g-1) \\ \varepsilon_i(g) & \text{otherwise,} \end{cases} \tag{B.3}$$

with CV_i the respective constraint violation linked to the ε_i parameter.

The above quoted strategy privileges optimality of solutions possibly at the expenses of their feasibility, due to the fact that $\varepsilon_0(g+1)$ in (B.2) is increasing in $F_{\ell_1}(x, \delta; \varepsilon(g+1))$ when the function value $f(x(g))$ increases. Moreover, to favour feasibility of solutions possibly at the expenses of their optimality, the penalty parameter $\varepsilon_i(g+1)$ in (B.3) is increased when the relative constraint violation in the g-th generation increases with respect to the previous one.

The procedure is also equipped by a splitting and refining technique for the positions of the particles. In particular, at each iteration, a particle p is split in its components $x^p(g)$ and $\delta^p(g)$ that are updated separately. For the vector $\delta^p(g)$ we employ the following updating rule

$$\delta_i^p(g+1) = \begin{cases} 1 & \text{if} \quad x_i^p(g) \in [l_i, u_i] \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, n$. Then, $\delta^p(g+1)$ is kept fixed and $F_{\ell_1}(x, \delta^p(g+1); \varepsilon(g+1))$ is minimized with respect to x, obtaining $\widetilde{x}^p(g+1)$. Finally, $\widetilde{x}^p(g+1)$ is refined getting

$$x_i^p(g+1) = \frac{\widetilde{x}_i^p(g+1)\delta_i^p(g+1)}{\sum_{i=1}^{n} \widetilde{x}_i^p(g+1)\delta_i^p(g+1)}, \tag{B.4}$$

for $j = 1, \ldots, NP$.

Appendix C. Nomenclature

Table 8
Table of notation.

Symbol	Description
n	Number of assets in the investable universe
x	$n \times 1$ vector of asset weights
R_i	Random variable representing the rate of return of asset i
μ_i	Expected value of R_i
$R_p(x)$	Random variable representing the rate of return of portfolio x
$\mu_p(x)$	Expected rate of return of portfolio x
C	$n \times n$ covariance matrix of the n stocks
$\sigma_p(x)$	Volatility (standard deviation) of portfolio x
$SR(x)$	Sharpe ratio of portfolio x
$MSR(x)$	Modified Sharpe ratio of portfolio x
k	Maximum number ($\leq n$) of assets included in the portfolio
$k_\%$	Fraction of assets making up the portfolio
δ	$n \times 1$ vector of binary variables denoting inclusion/exclusion from a portfolio
l_i	Lower bound for the i-th portfolio weight, $i = 1, \ldots, n$
u_i	Upper bound for the i-th portfolio weight, $i = 1, \ldots, n$
x_0	$n \times 1$ vector containing the current portfolio positions to rebalance
TR	Maximum turnover rate
NP	Number of candidate solutions in the swarm
MAX_{GEN}	Maximum number of generations
NL_g	Number of levels in generation g
LP_g	Number of particles in each level at generation g
$x^{l,p}(g)$	$n \times 1$ vector denoting the p-th particle in level L_l at generation g
$v^{l,p}(g)$	$n \times 1$ vector denoting the p-th velocity in level L_l at generation g
ϕ_g	Adaptive control parameter at generation g
$s(g)$	Swarm aggregation indicator at generation g
$t(g)$	Relative improvement between two consecutive generations
$\hat{x}^{l,p}(g)$	$n \times 1$ vector denoting the p-th mutated individual in level L_l at generation g
$\phi(x)$	Turnover constraint violation of candidate solution x
$F(x)$	Penalty function value for candidate solution x
CV_j	Constraint violation for the j-th constraint in the exact ℓ_1-penalty function approach
ε	6×1 vector of the exact ℓ_1-penalty function parameters
$F_{\ell_1}(x, \delta; \varepsilon)$	Exact ℓ_1-penalty function value for candidate solution x

Appendix D. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ins.2023.03.115.

References

[1] B.R. Auer, F. Schuhmacher, Performance hypothesis testing with the Sharpe ratio: the case of hedge funds, Finance Res. Lett. 10 (4) (2013) 196–208.

[2] H.J.C. Barbosa, A.C.C. Lemonge, H.S. Bernardino, A critical review of adaptive penalty techniques in evolutionary computation, in: Evolutionary Constrained Optimization, in: Infosys Science Foundation Series, Springer, New Delhi, 2015.

[3] A. Beck, First-Order Methods in Optimization, Society for Industrial and Applied Mathematics and Mathematical Optimization Society, Philadelphia, 2017.

[4] P. Beraldi, A. Violi, M. Ferrara, C. Ciancio, B.A. Pansera, Dealing with complex transaction costs in portfolio management, Ann. Oper. Res. 299 (1–2) (2021) 7–22.

[5] D. Bertsimas, R. Cory-Wright, A scalable algorithm for sparse portfolio selection, INFORMS J. Comput. 34 (3) (2022) 1489–1511.

[6] D. Bertsimas, R. Shioda, Algorithm for cardinality-constrained quadratic optimization, Comput. Optim. Appl. 43 (1) (2009) 1–22.

[7] F. Caraffini, F. Neri, M. Epitropakis, HyperSPAM: a study on hyper-heuristic coordination strategies in the continuous domain, Inf. Sci. 477 (2019) 186–202.

[8] N.A. Canakgoz, J.E. Beasley, Mixed-integer programming approaches for index tracking and enhanced indexation, Eur. J. Oper. Res. 196 (1) (2009) 384–399.

[9] M. Caporin, G. Jannin, F. Lisi, B. Maillet, A survey on the four families of performance measures, J. Econ. Surv. 28 (5) (2014) 917–942.

[10] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Trans. Cybern. 45 (2) (2015) 191–204.

[11] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inf. Sci. 291 (2015) 43–60.

[12] M. Corazza, G. di Tollo, G. Fasano, R. Pesenti, A novel hybrid PSO-based metaheuristic for costly portfolio selection problems, Ann. Oper. Res. 304 (2021) 109–137.

[13] M.F.P. Costa, R.B. Francisco, A.M.A.C. Rocha, E.M.G.P. Fernandes, Theoretical and practical convergence of a self-adaptive penalty algorithm for constrained global optimization, J. Optim. Theory Appl. 174 (2017) 875–893.

[14] Y. Crama, M. Schyns, Simulated annealing for complex portfolio selection problems, Eur. J. Oper. Res. 150 (3) (2003) 546–571.

[15] T. Cura, Particle swarm optimization approach to portfolio optimization, Nonlinear Anal., Real World Appl. 10 (4) (2009) 2396–2406.

[16] R. Das, D.K. Prasad, Prediction of porosity and thermal diffusivity in a porous fin using differential evolution algorithm, Swarm Evol. Comput. 23 (2015) 27–39.

[17] R. Datta, K. Deb, Evolutionary Constrained Optimization, Infosys Science Foundation Series, Springer, New Delhi, 2015.

[18] O. Ertenlice, C.B. Kalayci, A survey of swarm intelligence for portfolio optimization: algorithms and applications, Swarm Evol. Comput. 39 (2018) 36–52.

[19] J.B. Guerard, Handbook of Portfolio Construction. Contemporary Applications of Markowitz Techniques, Springer, New York, 2010.

[20] Y.C. Ho, D.L. Pepyne, Simple explanation of the no free lunch theorem of optimization, Cybern. Syst. Anal. 38 (2) (2002) 292–298.

[21] C. Israelsen, A refinement to the Sharpe ratio and information ratio, J. Asset Manag. 5 (2005) 423–427.

[22] M. Kaucic, Equity portfolio management with cardinality constraints and risk parity control using multi-objective particle swarm optimization, Comput. Oper. Res. 109 (2019) 300–316.

[23] M. Kaucic, F. Barbini, F.J. Camerota Verdù, Polynomial goal programming and particle swarm optimization for enhanced indexation, Soft Comput. 24 (2020) 8535–8551.

[24] P.N. Kolm, R. Tütüncü, F.J. Fabozzi, 60 years of portfolio optimization: practical challenges and current trends, Eur. J. Oper. Res. 234 (2014) 356–371.

[25] T. Krink, S. Mittnik, S. Paterlini, Differential evolution and combinatorial search for constrained index-tracking, Ann. Oper. Res. 172 (2009) 153–176.

[26] O. Ledoit, M. Wolf, Honey, I shrunk the sample covariance matrix, J. Portf. Manag. 30 (4) (2004) 110–119.

[27] K. Liagkouras, K. Metaxiotis, Examining the effect of different configuration issues of the multiobjective evolutionary algorithms on the efficient frontier formulation for the constrained portfolio optimization problem, J. Oper. Res. Soc. (2017).

[28] E. Mezura-Montes, C.A. Coello Coello, Constraint-handling in nature-inspired numerical optimization: past, present and future, Swarm Evol. Comput. 1 (4) (2011) 173–194.

[29] R. Moral-Escudero, R. Ruiz-Torrubiano, A. Suarez, Selection of optimal investment portfolios with cardinality constraints, IEEE Trans. Evol. Comput. (2006) 2382–2388.

[30] E.T. Oldewage, A.P. Engelbrecht, C.W. Cleghorn, The merits of velocity clamping particle swarm optimisation in high dimensional spaces, IEEE Symp. Ser. Comput. Intell. (2017) 1–8.

[31] E.T. Oldewage, A.P. Engelbrecht, C.W. Cleghorn, Movement patterns of a particle swarm in high dimensional spaces, Inf. Sci. 512 (2020) 1043–1062.

[32] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255.

[33] F. Schuhmacher, M. Eling, Sufficient conditions for expected utility to imply drawdown-based performance rankings, J. Bank. Finance 35 (9) (2011) 2311–2318.

[34] F. Schuhmacher, M. Eling, A decision-theoretic foundation for reward-to-risk performance measures, J. Bank. Finance 36 (7) (2012) 2077–2082.

[35] D.X. Shaw, S. Liu, L. Kopman, Lagrangian relaxation procedure for cardinality-constrained portfolio optimization, Optim. Methods Softw. 23 (3) (2008) 411–420.

[36] W. Shen, J. Wang, S. Ma, Doubly regularized portfolio with risk minimization, AAAI Conf. Artif. Intell. 28 (1) (2014) 1286–1292.

[37] K. Singh, R. Das, An experimental and multi-objective optimization study of a forced draft cooling tower with different fills, Energy Convers. Manag. 111 (2016) 417–430.

[38] G.-W. Song, Q. Yang, X.-D. Gao, Y.-Y. Ma, Z.-Y. Lu, J. Zhang, An adaptive level-based learning swarm optimizer for large-scale optimization, IEEE Trans. Syst. Man Cybern. (2021) 152–159.

[39] E.T. van Zyl, A.P. Engelbrecht, A subspace-based method for PSO initialization, IEEE Symp. Ser. Comput. Intell. (2015) 226–233.

[40] F. Wang, X. Wang, S. Sun, A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization, Inf. Sci. 602 (2022) 298–312.

[41] M. Woodside-Oriakhi, C. Lucas, J.E. Beasley, Heuristic algorithms for the cardinality constrained efficient frontier, Eur. J. Oper. Res. 213 (3) (2011) 538–550.

[42] Y. Xue, Y. Tong, F. Neri, An ensemble of differential evolution and Adam for training feed-forward neural networks, Inf. Sci. 608 (2022) 453–471.

[43] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspir. Comput. 2 (2) (2010) 78–84.

[44] Q. Yang, W.-N. Chen, J. Da Deng, Y. Li, T. Gu, J. Zhang, A level-based learning swarm optimizer for large-scale optimization, IEEE Trans. Evol. Comput. 22 (4) (2018) 578–594.

[45] J. Zhang, T. Leung, A. Aravkin, A relaxed optimization approach for cardinality-constrained portfolios, in: European Control Conference, Italy, 2019.

[46] H. Zhu, Y. Wang, K. Wang, Y. Chen, Particle Swarm Optimization (PSO) for the constrained portfolio optimization problem, Expert Syst. Appl. 38 (8) (2011) 10161–10169.