Annals of Operations Research

Editorial board Aims & scope Journal updates

The ***Annals of Operations Research*** publishes peer-reviewed original articles dealing with key aspects of operations research, including theory, practice, and computation. The journal publishes full-length research articles, short notes, expositions and surveys, reports on computational studies, and case studies that present new and innovative practical applications. — show all

For authors

Submission guidelines

About this journal

Electronic ISSN 1572-9338 **Print ISSN** 0254-5330

Abstracted and indexed in

ABS Academic Journal Quality Guide
ANVUR
Australian Business Deans Council (ABDC) Journal Quality List
BFI List
Baidu
CLOCKSS
CNKI
CNPIEC
Current Contents/Engineering, Computing and Technology
DBLP
Dimensions
EBSCO Academic Search
EBSCO Applied Science & Technology Source
EBSCO Business Source
EBSCO Computer Science Index
EBSCO Computers & Applied Sciences Complete
EBSCO Discovery Service
EBSCO Engineering Source
EBSCO STM Source
ECONIS
Gale
Google Scholar
INSPEC
Japanese Science and Technology Agency (JST)
Journal Citation Reports/Science Edition
Mathematical Reviews
Naver
Norwegian Register for Scientific Journals and Series
OCLC WorldCat Discovery Service
Portico
ProQuest ABI/INFORM
ProQuest Advanced Technologies & Aerospace Database
ProQuest-ExLibris Primo
ProQuest-ExLibris Summon
Research Papers in Economics (RePEc)
SCImago
SCOPUS
Science Citation Index
Science Citation Index Expanded (SCIE)
TD Net Discovery Service
UGC-CARE List (India)
Wanfang
zbMATH

Annals of Operations Research (2021) 304:109–137
https://doi.org/10.1007/s10479-021-04075-3

ORIGINAL RESEARCH

A novel hybrid PSO-based metaheuristic for costly portfolio selection problems

Marco Corazza[1] · Giacomo di Tollo[1] · Giovanni Fasano[2,3] · Raffaele Pesenti[2]

Accepted: 7 April 2021 / Published online: 21 April 2021

Abstract
In this paper we propose a hybrid metaheuristic based on Particle Swarm Optimization, which we tailor on a portfolio selection problem. To motivate and apply our hybrid metaheuristic, we reformulate the portfolio selection problem as an unconstrained problem, by means of penalty functions in the framework of the exact penalty methods. Our metaheuristic is hybrid as it adaptively updates the penalty parameters of the unconstrained model during the optimization process. In addition, it iteratively refines its solutions to reduce possible infeasibilities. We report also a numerical case study. Our hybrid metaheuristic appears to perform better than the corresponding Particle Swarm Optimization solver with constant penalty parameters. It performs similarly to two corresponding Particle Swarm Optimization solvers with penalty parameters respectively determined by a REVAC-based tuning procedure and an *irace*-based one, but on average it just needs less than 4% of the computational time requested by the latter procedures.

Keywords Hybrid metaheuristics · Particle Swarm Optimization · Global optimization · Portfolio selection problems · Exact penalty functions · REVAC · *irace*

✉ Giacomo di Tollo
giacomo.ditollo@unive.it

Marco Corazza
corazza@unive.it

Giovanni Fasano
fasano@unive.it

Raffaele Pesenti
pesenti@unive.it

[1] Department of Economics, Università Ca' Foscari, Venezia, Sestiere di Cannaregio 873, 30121 Venezia, Italy

[2] Department of Management, Ca' Foscari University of Venice, Sestiere di Cannaregio 873, 30121 Venezia, Italy

[3] National Research Council – Maritime Technology Research Institute (CNR – INSEAN), Via di Vallerano 139, 00128 Rome, Italy

1 Introduction

Setting the parameters used within an algorithm is a key-point to insure its reliability, performances, robustness, and scalability. Although many approaches resort to experts' judgement to determine the algorithm's parameter values (see Kotthoff et al. 2019), the literature proposes a great number of parameter setting procedures (Lobo et al. 2007). As in Eiben et al. (1999), we can partition these approaches in *parameter tuning techniques* (also referred to as *off-line configuration*), which determine the algorithm parameters values before the algorithm execution, and *parameter control techniques* (also referred to as *on-line control*), which continuously update the parameter values during the algorithm execution.

On this guideline, also Particle Swarm Optimization (PSO) has been used to assess the parameters of other algorithms. In this regard we have for instance: (a) (Hong 2009), where parameters value for a Support Vector Regression model are determined, using chaotic PSO, (b) (Lin et al. 2008), where PSO is used to set parameters for Support Vector Machines, (c) (Si et al. 2012) that uses PSO to tune Differential Evolution parameters.

Conversely, several approaches have also been proposed in the literature to determine PSO parameters value. These approaches get started from extensive studies on PSO parameters (inertia weight and coefficients), since the early PSO related research (Clerc and Kennedy 2002; Eberhart and Shi 2001; Shi and Eberhart 1998a, b). In this context, Trelea (2003), Campana et al. (2010) study the possible range for PSO parameters in order to evaluate their impact on convergence.

Methodologies and concepts to determine PSO parameter values can be partitioned in *tuning* and *control* methods. Our contribution can be framed in this latter class of methods that in the PSO jargon are also referred as to *adaptive*.

Amongst parameter *tuning* procedures, Dai et al. (2011) proposes the idea of using an additional PSO scheme that analyses the impact of each PSO parameter, while Wang et al. (2014) proposes to use Taguchi method. In addition, other general purpose procedures of this type could also be applied to PSO such as: (1) statistical procedures to evaluate parameter settings and to eliminate candidate parameters configurations that are dominated by others (Trujillo et al. 2020; Birattari et al. 2010); (2) meta-heuristic methods to explore the candidate configurations space (Nannen et al. 2008; Hutter et al. 2007); (3) sequential model-based optimisation in order to define both a correlation between parameter settings and algorithm performance, and to identify high-performing parameter values (Hutter et al. 2011); (4) other approaches, including Bayesian Optimization (Eggensperger et al. 2013), jointly used with Gaussian process (Snoek et al. 2012), Random Forests (Hutter et al. 2011), and Tree Parzen Estimator (Bergstra et al. 2011) (see Huang et al. 2019 for a detailed overview of parameter tuning approaches).

Generally speaking, parameter tuning may be time consuming: this is why tuning is often done by using *cheap* synthetic test functions that may turn to be rather different from the real benchmarks, or by using *cheap-to-evaluate* surrogates of real hyperparameter optimization benchmarks (Eggensperger et al. 2015).

Amongst *control* procedures we find: Shi and Obaiahnahatti (1998), which presents a basic adaptive procedure for the assessment of PSO parameters that makes the inertia weight decrease linearly over time; Zhan and Zhang (2008), which introduces the Adaptive Particle Swarm Optimization (APSO) that defines four evolutionary states to control the inertia weight and the acceleration coefficients (along with other parameters); Hsieh et al. (2009), which proposes an adaptive population management procedure to automatically determine the population size; Winner et al. (2009), which employs non-explicit control parameters that

describe self-organizing systems at an abstract level; Tang et al. (2011), which uses the search history collected by particles to determine acceleration coefficients; time-varying acceleration coefficients are considered also in Ratnaweera et al. (2004a). Stemming algorithms derived from Genetic and Evolutionary Algorithms can be also seen as *control* procedures for PSO. As an example, this is the case when mutation operators are introduced to avoid premature convergence, as suggested by many contributions (Si et al. 2011; Sharma and Chhabra 2019; Jana et al. 2019; Wang et al. 2019). Recently, a mechanism to control the balance between exploration and exploitation has been detailed in Xia et al. (2020) (Dynamic Multi-Swarm Global Particle Swarm Optimization), and a great attention to define learning strategies to increase swarm diversity was given in Zhang et al. (2020). The interested reader can find a comparative analysis among PSO schemes in, e.g., Harrison et al. (2018) where 18 different self-adaptive PSO algorithms are investigated.

Adaptive versions of PSO (Zhan et al. 2009) have been applied to a plethora of problems, see (Marinakis et al. 2015) for a literature review.

As regards the application of PSO techniques to portfolio optimization problems, some cares and preliminaries are mandatory. Making effective decisions in real economic and financial contexts may imply having to deal with complex or even NP-hard mathematical programming problems (see, e.g., Arora et al. 2011). The modeling of many economic and financial systems is not straightforward, and it may need to resort to non-analytical functions or to a mixed-integer framework. In addition, on one hand, it requires to take into account uncertainty, which is congenital to the economic environments. On the other hand, professional operators may find difficult to use cumbersome models that require excessive computational power. They may prefer to settle to extremely simplified decision models even when they provide "solutions" that are fairly far from the optimal ones.

In the last decades, the above reasons and the greater availability of computational power have fostered an increasing interest towards the development and the applications of metaheuristics. The interested reader is forwarded, as an example, to Soler-Domínguez et al. (2017) that reports the increasing number of papers on applications of metaheuristics to finance since 2000.

In this paper we propose a novel hybrid metaheuristic based on PSO, for approximately solving complex mathematical programming problems as those introduced above. In particular, we tailor this hybrid metaheuristic on the portfolio selection problem presented in Corazza et al. (2013). This problem is in general NP-hard, and its objective and constraints are both nondifferentiable and nonconvex. We solve it using an exact penalty method which transforms the constrained problem into an unconstrained one.

Our metaheuristic mainly consists of a PSO module and of other *hybridizing procedures*. The former one jointly minimizes both the original objective function and all the constraint violations. The latter ones initialize the solution search, adaptively update the penalty parameters and, finally, are used to refine the obtained solution.

We compare the results obtained by our hybrid metaheuristic with those provided by three PSO-based solvers. In the first solver, the penalty parameters are simply kept constant, as often done in the literature (see, e.g., Corazza et al. 2013). In the second and in the third solver the penalty parameters are a-priori determined by a REVAC-based tuning procedure and an *irace*-based one, respectively (see, for details, Nannen and Eiben 2007a; López-Ibáñez et al. 2016). Our hybrid metaheuristic appears to perform better than the first PSO-based solver, while it seems to perform similarly both to the second and to the third PSO-based solver. However, our hybrid metaheuristic just needs on average less than 4% of the computational time requested by the latter PSO-based solvers. In particular, all such evidences hold even when a reduced number of iterations is allowed for the solvers, e.g., in case of optimization

problems for which computing the value of the solution is costly. This makes our hybrid metaheuristic a flexible tool that can provide a fast approximate solution to a financial expert, who frequently needs to select portfolios in real time. We observe that there is also the chance to preliminary propose an *offline* application of REVAC/*irace*, over a given prototype problem, and then to use the resulting computed PSO parameters on the current instance. Nevertheless, this approach implies a couple of drawbacks: the resulting PSO parameters, to be used on the current problem, would be just suboptimal; moreover, there is no guarantee that the problem used for PSO tuning parameters has a comparable complexity with respect to the problem in hand. Both the last issues unavoidably risk to deteriorate the performance of PSO on the current problem.

In the next sections we provide both methodological motivations and numerical results that reveal why our hybrid metaheuristic shows faster progresses, since the early iterations, than classical PSO-based approaches. In particular, we argue that the structure of the considered portfolio optimization problem, along with the fact that only its fast approximate solution is sought, suggested our choice for a dynamic (say *adaptive*) penalty approach (see also Sects. 4.1 and 5). As regards the last issue, we refer the interested reader to Griffin and Kolda (2010). This study presents possible guidelines for approximately solving complex constrained optimization problems, when differentiability is not a mandatory issue for the penalty framework.

Our preference for a PSO-based solver, with respect to other possible alternative heuristics, relies also on the results in Corazza et al. (2012), where the use of Genetic Algorithms for solving a similar portfolio problem was investigated, and a PSO approach appeared to perform better. Some other alternatives were also considered such as: Filter Methods (Nocedal and Wright 2006), Augmented Lagrangian Methods (Nocedal and Wright 2006) or Lagrangian Relaxation (Fisher 1985). However, they were excluded as they seemed to fit less our efficiency requisites than a PSO approach, as we argue at the end of Sect. 3.

For the sake of completeness, as regards portfolio selection problems, we also refer the reader to the landmark papers Konno and Wijayanayake (1999) and Konno and Yamamoto (2005), which focus on a theoretical approach issuing both a specific measure of risk and transaction costs. Finally, the more recent extensions of PSO-based approaches for portfolio selection problems in Chen and Zhang (2010) and Ray and Klepac (2019) are worth investigating.

On balance, the main contributions of this paper, along with its elements of novelty with respect to the current literature, can be summarized as follows:

- For our mixed-integer formulation of the portfolio selection problem we draw inspiration from the penalty approach in Corazza et al. (2013, 2012, 2019). However, unlike the latter references, we split the procedure to update some subsets of variables in the problem, in order to better exploit convexity with respect to a restricted number of unknowns.
- With respect to Corazza et al. (2013, 2012) we adopt an adaptive (dynamic) update of penalty parameters, pursuing a twofold purpose. First we aim at preserving theoretical issues on penalty methods for nonsmooth problems, then our settings are committed to yield convincing numerical performance (see Sect. 4.1).
- With respect to Corazza et al. (2013, 2012, 2019), in our framework we embed a procedure to update some of the problem unknowns, in accordance with the idea of *Schwarz Alternating Methods (SAM)* (Gander 2008); i.e., we first split and then we refine the vector of PSO particles' position (see Sect. 4.2).
- This paper proposes a complete numerical experience (which is first intended to complement and then to extend the one in Corazza et al. (2013)). Moreover, our approach is also

compared with both state-of-the-art software (namely REVAC and *irace*) for *parameter tuning*, and an exact method for mixed-integer programming problems (see Appendix A).

The remainder of this paper is organized as follows. In the next section, we recall the basics of PSO. In Sect. 3, we present the portfolio selection problem used as a reference problem throughout this paper. In Sect. 4, we introduce our hybrid metaheuristic. In Sect. 5, we describe the plan of our numerical experience and the issues that can arise. Then, we report the results obtained from the application of the different metaheuristics. In Sect. 6, we draw some final remarks.

The paper includes also an appendix, where we present the formulation of the portfolio selection problem as a standard nonlinear mixed-integer programming problem. We use this model to have reference exact solutions and to assess the approximation errors of the solutions provided by our hybrid metaheuristic.

2 Basics on PSO

PSO is a metaheuristic iterative method for the solution of global optimization problems (Kennedy and Eberhart 1995). It belongs to the class of bio-inspired methods which attempt to emulate some natural paradigms of behavior, related to groups of individuals. Examples of similar techniques can be found in the comprehensive study (Talbi and Nakib 2019), showing their efficiency. PSO iteratively attempts to replicate the rationale behind a swarm foraging for food. Each member of the swarm is called *particle*. Several PSO variants have been proposed in the literature, both for unconstrained and constrained problems (Wu and Zhang 2013; Liang and Suganthan 2006), their performances depending often on the function to optimize and the shape of its level sets.

Let $P \in \mathbb{N}$ be the size of the swarm, $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function to minimize, also referred to as *fitness function* in the PSO jargon. We assume that the level set

$$\mathcal{L}_f(\bar{y}) = \{y \in \mathbb{R}^n \ : \ f(y) \le f(\bar{y})\}$$

is compact, for any given vector $\bar{y} \in \mathbb{R}^n$, so that the minimization problem

$$\min_{y \in \mathbb{R}^n} f(y) \tag{1}$$

surely admits global solutions.

At iteration k of PSO, the *position* $y_j^k \in \mathbb{R}^n$ of each particle j of the swarm represents a *tentative solution* for (1). Then, the j-th particle updates its position according with the rule

$$y_j^{k+1} = y_j^k + v_j^{k+1}, \qquad j = 1, \ldots, P, \ k = 0, 1, \ldots,$$

being $y_j^{k+1} \in \mathbb{R}^n$ its next position (tentative solution), while $v_j^{k+1} \in \mathbb{R}^n$ is its *velocity*, i.e., the search direction at y_j^k.

The direction v_j^{k+1} is typically computed as the cone combination of three contributions. Namely, setting

$$p_j^k \in \arg\min_{0 \le h \le k} \left\{ f(y_j^h) \right\}, \qquad j = 1, \ldots, P, \quad k = 0, 1, \ldots, \tag{2a}$$

$$p_g^k \in \arg\min_{\substack{0 \le h \le k \\ j = 1, \ldots, P}} \left\{ f(y_j^h) \right\}, \qquad k = 0, 1, \ldots, \tag{2b}$$

where the vector p_j^k, respectively p_g^k, is the best solution so far found by particle j and by the swarm, the search direction v_j^{k+1} is given by Kennedy and Eberhart (1995)

$$v_j^{k+1} = v_j^k + \alpha_j^k \otimes (p_j^k - y_j^k) + \beta_j^k \otimes (p_g^k - y_j^k), \qquad j = 1, \ldots, P, \quad k = 0, 1, \ldots, \tag{3}$$

being:

- the vector v_j^k the so called *inertia* of particle j to change trajectory;
- the vector $p_j^k - y_j^k$ the deviation of y_j^k from the best previous position of particle j,
- the vector $p_g^k - y_j^k$ the deviation of y_j^k from the best previous solution so far found by the swarm.

Finally, $\alpha_j^k, \beta_j^k \in \mathbb{R}^n$ are positive random vectors, while the symbol '$\otimes$' indicates the entry-by-entry product between vectors. In the literature, parameter α_j^k is often addressed as the *cognitive parameter*, while β_j^k as the *social parameter*. In addition, they are usually expressed as:

$$\alpha_j^k = c_j^k r_1^k, \qquad j = 1, \ldots, P, \; k = 0, 1, \ldots,$$
$$\beta_j^k = c_g^k r_2^k, \qquad j = 1, \ldots, P, \; k = 0, 1, \ldots,$$

where, for any j and k, r_1^k and r_2^k are n-real vectors whose entries are determined according to the prominent literature, while c_j^k, c_g^k entries assume values as described in Sect. 5.

We remark that, unlike the standard gradient based methods, the search direction v_j^{k+1} is not necessarily a descent direction for the function f at y_j^k. The new position y_j^{k+1} that the j-th particle computes at step k might not improve the objective function value, though it might prevent the solutions to be entrapped in a neighborhood of a local minimum. Indeed, the update (3) is designed to perform both an *exploration* and an *exploitation* in $\mathbb{R}^n$. The vector $\beta_j^k \otimes (p_g^k - y_j^k)$ is mainly responsible for exploration, i.e., for the search of global minima over the entire feasible set, avoiding entrapment in neighborhoods of poor local minima. The vector $\alpha_j^k \otimes (p_j^k - y_j^k)$ is mainly responsible for exploitation, i.e., for refining the solutions nearby promising local minima, when no further progress from exploration is experienced.

In this paper, in accordance with Ozcan et al. (2016), we consider the following slightly more general expression for the velocity:

$$v_j^{k+1} = \chi^k \left[w^k v_j^k + \alpha_j^k \otimes (p_j^k - y_j^k) + \beta_j^k \otimes (p_g^k - y_j^k) \right], \qquad j = 1, \ldots, P, \; k = 0, 1, \ldots, \tag{4}$$

where χ^k and w^k are positive parameters (see also Sect. 5.3 for the choice of their values) respectively known as the *constriction coefficient* and *inertia coefficient*.

Important contributions have been recently published, which ensure that by a proper choice of the coefficients in (4), some necessary conditions of convergence for PSO iteration can be given. We refer the interested reader to, e.g., (Clerc and Kennedy 2002; Campana et al. 2010; Bonyadi and Michalewicz 2016).

3 Our reference portfolio selection problem

Broadly speaking, a portfolio selection problem consists in choosing a subset of assets with the purpose of obtaining an appreciable return while keeping risk at a reasonable level.

Developing portfolio selection models for real stock markets is in general a complex task for several reasons. As an example effective models are often asked to:

- gauge the uncertainty by adopting risk measures that both satisfy appropriate formal properties (e.g., coherence) and cope with the generally non-normal return distributions of the real stock markets (Artzner et al. 1999). Risk measures should be designed to take into account the risk attitude of different investors;
- provide a certain number of possible alternatives, when requested by the investors that desire to make their final choice assessing the outcome of different scenarios;
- take into account several practices and rules of the portfolio management industry that may affect the portfolio selection process. For instance, in the standard professional practice, the fund managers self-impose bounds on the minimum and the maximum number of assets to trade, in order to control the transaction costs;
- provide fast and reliable approximate solutions, rather than accurate but time consuming ones. This holds in particular when the return of the approximate proposal is not significantly different with respect to an exact (time consuming) one.

3.1 The constrained model

In this paper, we start from considering the portfolio selection model proposed in Corazza et al. (2013, 2019). This model adopts a coherent risk measure based on the combination of lower and upper partial moments of different orders of the portfolio return distribution. This measure can manage non-Gaussian distributions of asset returns and can reflect different investors' risk attitudes (Chen and Wang 2008). It takes into account both the risk contained in the "bad" tail (the left one of the portfolio return), and the advantages of using the "good" tail (the right one of the same portfolio return), see, e.g., Artzner et al. (1999) for further details. The considered model includes *cardinality constraints* to bound the minimum and the maximum number of assets to trade, and includes also constraints on the minimum and the maximum capital percentage to invest in each asset. These constraints often result from a matching between broker's knowledge and investor's requests.

Before formalizing the portfolio selection problem of interest, we need to introduce the notations which follow:

- Parameters:
  - N: number of possible investment assets;
  - r_e: minimum desired expected return of the portfolio;
  - K_d and K_u: minimum and maximum number of assets to trade, respectively;
  - d and u: minimum and maximum capital percentage to invest in each asset, respectively;
  - p: index of the norm used in the risk measure of the portfolio, with $p \geq 1$, representing investor's attitude to risk;
  - a: relative weight ($0 \leq a \leq 1$) assigned in the risk measure of the portfolio to the good tail of the portfolio return distribution, with respect to the bad tail;
  - r_i: (stochastic) return of the i-th asset, for $i = 1, \ldots, N$.
- Decisional variables:
  - x_i: continuous variable expressing the percentage of the portfolio invested in the i-th asset, for $i = 1, \ldots, N$;

– z_i: indicator variable assuming value 1 if the i-th asset is included in the portfolio, 0 otherwise, for $i = 1, \ldots, N$.

In addition, x and z indicate respectively the N-dimensional vectors $(x_1, \ldots, x_N)^T$ and $(z_1, \ldots, z_N)^T$ and:

- $E[y]$ indicates the expected value of the random argument y,
- y^- indicates $\max\{0, -y\}$;
- y^+ indicates $(-y)^-$;
- $\widehat{r}$ stands for the vector $(\widehat{r}_1, \ldots, \widehat{r}_N)^T$ of expected values $\widehat{r}_i = E[r_i]$, $i = 1, \ldots, N$.

Note that hereinafter we denote the i-th entry of a vector s by either s_i or $(s)_i$, the latter notation is used if otherwise interpretation ambiguities may arise. Moreover, if $u, v \in \mathbb{R}^N$ then $u \leq v$ $[u < v]$ is equivalent to the N inequalities $u_i \leq v_i$ $[u_i < v_i]$, $i = 1, \ldots, N$.

Given the above notation, we can express the overall stochastic portfolio return as $r = \sum_{i=1}^{N} r_i x_i$, and consequently the expected portfolio return as

$$E[r] = \sum_{i=1}^{N} \widehat{r}_i x_i.$$

Accordingly, we express the risk measure of the portfolio return as

$$\rho_{a,p}(r) = a\|(r - E[r])^+\|_1 + (1-a)\|(r - E[r])^-\|_p - E[r].$$

The risk measure $\rho_{a,p}(r)$ is coherent, as proved in Chen and Wang (2008), and allows to describe the investor's risk attitude through an appropriate tuning of the non-negative values of parameters a and p.

Following the notation of the authors in Chen and Wang (2008), we are now ready to formulate the portfolio selection problem as follows:

$$\min_{x,z} \ \rho_{a,p}(r) = a\|(r - E[r])^+\|_1 + (1-a)\|(r - E[r])^-\|_p - E[r] \tag{5a}$$

$$\text{s.t.} \ \ E[r] \geq r_e \tag{5b}$$

$$\sum_{i=1}^{N} x_i = 1 \tag{5c}$$

$$K_d \leq \sum_{i=1}^{N} z_i \leq K_u \tag{5d}$$

$$z_i d \leq x_i \leq z_i u, \qquad i = 1, \ldots, N \tag{5e}$$

$$z_i \in \{0, 1\}, \qquad i = 1, \ldots, N. \tag{5f}$$

Constraint (5b) imposes the minimum desired expected return of the portfolio. Constraint (5c) imposes a budget constraint. Constraints (5d) and (5e) impose respectively bounds on the number of assets traded and on the capital percentage to invest in each asset of the portfolio. In particular, the left inequality in (5e) suggests that short-selling is not allowed, as long as $d \geq 0$. Finally, constraints (5f) impose that an asset is either included or excluded from the portfolio, i.e. the variables z_i, $i = 1, \ldots, N$ are binary. In the next section we give a framework for the transformation of (5) into an unconstrained optimization problem, so that PSO can be applied for its approximate solution.

3.2 An unconstrained model

Here, we show how to reformulate (5) as an unconstrained optimization problem by means of penalty functions, so that PSO can be applied. To this end, initially we recall some basic results on penalty functions.

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, and two vector functions $h = \{h_1, h_2, \ldots, h_m\} : \mathbb{R}^n \to \mathbb{R}^m$ and $g = \{g_{m+1}, g_{m+2}, \ldots, g_p\} : \mathbb{R}^n \to \mathbb{R}^{p-m}$, with $f, h, g \in \mathcal{C}^1(\mathbb{R}^n)$, consider the constrained optimization problem

$$\min_{y \in \mathcal{F}} f(y), \tag{6}$$

where $\mathcal{F} = \{y \in \mathbb{R}^n \ : \ h(y) = 0, \ g(y) \leq 0\}$ is compact.

We can associate to (6) the following ℓ_1-nondifferentiable exact *penalty function* (Zangwill 1967)

$$P(y; \eta) = f(y) + \sum_{i=1}^{m} \frac{1}{\eta_i} \|h_i(y)\|_1 + \sum_{j=m+1}^{p} \frac{1}{\eta_j} \left\|\max\{0, g_j(y)\}\right\|_1 \tag{7}$$

being $\eta = \{\eta_1, \eta_2, \ldots, \eta_m, \eta_{m+1}, \ldots, \eta_{p-1}, \eta_p\} > 0$ a vector of positive penalty parameters.

Then, the *Mangasarian-Fromowitz Constraint Qualification* (MFCQ) (Bazaraa et al. 2006) condition holds at point $\hat{y} \in \mathcal{F}$ for problem (6) if:

(a) the vectors $\nabla h_1(\hat{y}), \ldots, \nabla h_m(\hat{y})$ are linearly independent;
(b) there exists a nonzero vector $d \in \mathbb{R}^n$ such that

1. $\nabla h_i(\hat{y})^T d = 0$, for $i = 1, \ldots, m$
2. $\nabla g_j(\hat{y})^T d < 0$, for $j = m + 1, \ldots, p$ and $g_j(\hat{y}) = 0$.

Finally, consider the following unconstrained minimization problem

$$\min_{y \in \mathring{\Omega}} P(y; \eta). \tag{8}$$

where $\mathring{\Omega}$ is an open set that contains the compact set $\mathcal{F}$, (i.e., $\mathring{\Omega} \supset \mathcal{F}$). In addition, denote with Ω a closure of the open set $\mathring{\Omega}$ (i.e., $\Omega = \mathrm{cl}(\mathring{\Omega})$). The following proposition holds (see, e.g., Mangasarian and Han 1979).

Proposition 1 *Consider the problems* (6) *and* (8). *If*

- *MFCQ holds at any global minimum of P* (6),
- *there exists a set* Ω *such that* $\Omega = \mathrm{cl}(\mathring{\Omega})$ *and* $\mathring{\Omega} \supset \mathcal{F}$

then, there exists a vector $\eta^* > 0$ *such that for any* $\eta \in (0, \eta^*]$, *any global minimum of* (6) *is a global minimum of* (8) *and vice versa.*

Proposition 1 establishes a relation between the solutions of (6) and (8). In particular, it implies that constrained problem (6) can be solved by efficient iterative descent methods for the unconstrained problem (8). However, in general, iteratively solving problem (8) starting from an initial point $\bar{y}$ and a given choice of the real parameter $\bar{\eta} > 0$, may not yield a solution of (6), because the level set

$$\mathcal{L}(P, \bar{y}, \bar{\eta}) = \{y \in \mathbb{R}^n \ : \ P(y; \bar{\eta}) < P(\bar{y}; \bar{\eta})\} \tag{9}$$

possibly does not satisfy the condition $\mathcal{L}(P, \bar{y}, \bar{\eta}) \supseteq \mathcal{F}$, as implicitly required by the second condition in Proposition 1.

The above considerations motivate our proposal for adaptively updating the penalty parameters. Indeed, the choices of both Ω and η are crucial for the possibility of determining an optimal point of (6) by solving (8). Unfortunately, neither Ω nor η can be usually known a-priori. For example, we might be induced to set η very small. However, we could provide no guarantee that $\eta \leq \eta^*$ holds. In addition, if η is too small, serious ill-conditioning might arise, implying numerical instability and possibly slow progress at each iteration of a descent solution method.

We decided to adopt for our reference portfolio selection problem a standard exact penalty framework, given its simplicity and since it guarantees sufficient theoretical results under mild assumptions. Nevertheless, we cannot exclude that other penalty approaches could suit better when $f(y)$ is non-differentiable.

To apply the results in Proposition 1 to our portfolio selection problem (5), we first have to replace the constraint $z_i \in \{0, 1\}$ (i.e. (5f)) with $z_i(1 - z_i) = 0$, with $i = 1, \ldots, N$. In this way, we obtain that the feasible set of (5) is surely compact.

Unfortunately, point (a) of the MFCQ condition might not be satisfied at some feasible points. In addition, function $\rho_{a,p}(r)$ in (5a) is not continuously differentiable as required. All the same, we can still adopt a penalty framework by invoking the general result in Bazaraa et al. (2006)-Theorem 9.22, which requires only the continuity of the objective function, although convergence properties are partially lost.

In particular, we set $\Omega = \mathbb{R}^{2N}$ and adaptively update the vector of parameters η, accepting the possibility that some of its entries approach very small values (see Sect. 5.1). As some convergence results are partially lost, we will introduce further corrections to improve performance. Considering (5), our ℓ_1-penalty problem becomes

$$\min_{x \in \mathbb{R}^N,\ z \in \mathbb{R}^N} P(x, z; \varepsilon) \tag{10}$$

with $\varepsilon \in \mathbb{R}^8$ and

$$\begin{aligned} P(x, z; \varepsilon) = \rho_{a,p}(r) + \frac{1}{\varepsilon_0} \Bigg[& \varepsilon_1 \max\left\{0, r_e - \sum_{i=1}^{N} \hat{r}_i x_i\right\} + \varepsilon_2 \left| \sum_{i=1}^{N} x_i - 1 \right| \\ & + \varepsilon_3 \max\left\{0, K_d - \sum_{i=1}^{N} z_i\right\} + \\ & + \varepsilon_4 \max\left\{0, \sum_{i=1}^{N} z_i - K_u\right\} + \varepsilon_5 \sum_{i=1}^{N} \max\{0, z_i d - x_i\} + \\ & + \varepsilon_6 \sum_{i=1}^{N} \max\{0, x_i - z_i u\} + \varepsilon_7 \sum_{i=1}^{N} |z_i(1 - z_i)| \Bigg] \end{aligned}$$

where $\varepsilon = (\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_7)^T > 0$. We remark that each of the penalty parameters η_k in (7) is replaced by a ratio of parameters $\varepsilon_k/\varepsilon_0$ in $P(x, z; \varepsilon)$. This choice is motivated by efficiency reasons, as clarified in the next sections.

The existence of a unique minimizer of $P(x, z; \varepsilon)$ is not guaranteed. Hence, the necessity to tackle the problem (10) by a global method. Finally, considering that even the problem (10) is in general NP-hard, and that practitioners may need a fast approximate solution of their portfolio problems to compare different scenarios, we decided to move away the focus of the paper from asymptotically convergent exact global methods, when solving (10). In this regard, our choice of adopting PSO seems to provide a reasonable compromise between

precision of the approximate solution in the early iterations, and computational burden, as the numerical results in Sect. 5 seem to confirm.

We remark that other possible alternative approaches to approximately solve (10) can be considered. Among them we find Lagrangian Relaxation methods (see for instance Fisher 1985; Bertsekas 2016), which can also provide appealing bounds on the value of the objective function. In particular, they consist of moving (dualizing) some inequality constraints to the objective function, after multiplying them by some nonnegative values (dual variables). This approach proved to work efficiently on several classes of optimization problems, both linear and nonlinear. However, on our portfolio optimization problem, the iterative computation of the dual variables might require a cumbersome and possibly inefficient updating procedure. The PSO choice to assess the penalty parameters, based on the knowledge available at the current iteration, appeared more efficient.

4 Our hybrid metaheuristic

In this section we describe our hybrid metaheuristic, hereafter referred also as PSO-D (*D* stands for *dynamic*). Its pseudo-code is reported by the Algorithm 1. The metaheuristic includes an initialization phase and an iteration phase. In turn, the iteration phase includes an external and an internal loop. The values of the position of the particles, i.e., of variables x_j and z_j, for $j = 1, \ldots, P$, are updated in the internal loop. The value of the penalty parameter vector ε is updated in the external loop.

Our hybrid metaheuristic includes two distinctive characteristics: the adaptive change of the penalty parameter vector ε, and the split and refinement of the particle positions, in addition to their updating.

4.1 Penalty parameter vector ε settings and updating

Assessing effective penalty parameters is a tricky issue and depends on the class of problems under concern. We propose an adaptive tuning of these parameters based on the overall progress of the metaheuristic.

Hereinafter, we use the symbol ε^k to indicate the value of vector ε at iteration k. In addition, in accordance with what we have defined in (2b), we indicate p_g^k as the best position among PSO particles until iteration k. In particular, we split and express p_g^k (similarly for p_j^k) as

$$p_g^k = (x_g^k, z_g^k)$$

to emphasize that a particle position has two subvectors of components, the subvector of variables x^k and the subvector of variables z^k.

For $k = 0$ the initial parameters vector ε^0 is set as

$$\varepsilon^0 = (\varepsilon_0^0, \varepsilon_1^0, \varepsilon_2^0, \varepsilon_3^0, \varepsilon_4^0, \varepsilon_5^0, \varepsilon_6^0, \varepsilon_7^0) = (10^{-4}, 1, 1, 1, 1, 1, 1, 1) \in \mathbb{R}^8.$$

The values of the entries $\varepsilon_i^0, i = 1, \ldots, 7$ are chosen to initially impose an equal penalization for all constraints violations. Differently, the value of ε_0^0 is chosen much smaller in order to initially privilege feasible solutions.

For $k \geq 1$, the vector ε^k is updated as follows. We update ε_0^k by checking for a possible decrease of the value of $\rho_{a,p}(r_g^k)$, where $r_g^k = \sum_{i=1}^N \left(x_g^k\right)_i r_i$. For $i = 1, \ldots, 7$, we update ε_i^k,

Algorithm 1: Pseudo-code of hybrid metaheuristic PSO- D(Π) that returns a (sub)-optimal solution of portfolio selection problem Π.

PSO- D(Π)
Input: Π a constrained portfolio selection problem of type (5)
Output: A (sub)-optimal solution $y^* = (x^*, z^*)$ to problem Π

Initialization:
begin
  reformulate Problem (5) into Problem (8)
  set initial value of $\varepsilon^0 = \{\varepsilon_0^0, \varepsilon_1^0, \varepsilon_2^0, \ldots, \varepsilon_7^0\}$
  foreach *particle* j **do**
    set initial values x_j^0
  end
  set $k = 0$
end

Iteration k:
repeat
  repeat
    foreach *particle* j **do**
      set the vector z_j^{k+1}
      update the vector x_j^{k+1}
      refine the vector x_j^{k+1}
    end
    $k = k + 1$
  until *internal loop STOP condition*
  update the value of ε^k
until *external loop STOP condition*
return (x_g^k, z_g^k).

by checking for the violation χ_i of the i-th constraint:

$$\chi_1(x_g^k, z_g^k) = \max\left\{0, r_e - \sum_{i=1}^{N} \hat{r}_i \left(x_g^k\right)_i\right\}$$
$$\chi_2(x_g^k, z_g^k) = \left|\mathbf{1}^T x_g^k - 1\right|$$
$$\chi_3(x_g^k, z_g^k) = \max\{0, K_d - \mathbf{1}^T z_g^k\}$$
$$\chi_4(x_g^k, z_g^k) = \max\{0, \mathbf{1}^T z_g^k - K_u\}$$
$$\chi_5(x_g^k, z_g^k) = \sum_{i=1}^{N} \max\left\{0, \left(z_g^k\right)_i d - \left(x_g^k\right)_i\right\}$$
$$\chi_6(x_g^k, z_g^k) = \sum_{i=1}^{N} \max\left\{0, \left(x_g^k\right)_i - \left(z_g^k\right)_i u\right\}$$
$$\chi_7(x_g^k, z_g^k) = \sum_{i=1}^{N} \left|\left(z_g^k\right)_i \left(1 - \left(z_g^k\right)_i\right)\right|.$$

We also adopted the following strategy:

- Every 5 iterations of the PSO-D internal loop, we update the entry ε_0^k of ε^k, according with the following rule:

$$\varepsilon_0^{k+1} = \begin{cases} \min\left\{3\varepsilon_0^k, 1\right\} & \text{if } \rho_{a,p}(r_g^k) \geq \rho_{a,p}(r_g^{k-1}) \\ \max\left\{0.6\varepsilon_0^k, 10^{-15}\right\} & \text{if } \rho_{a,p}(r_g^k) < 0.90 \cdot \rho_{a,p}(r_g^{k-1}) \\ \varepsilon_0^k & \text{otherwise.} \end{cases} \tag{11}$$

  In all the other iterations, $\varepsilon_0^{k+1} = \varepsilon_0^k$.
- Every 10 iterations of the PSO-D internal loop, we update the entries ε_i^k, $i = 1, \ldots, 7$, of ε^k, according with the following rule:

$$\varepsilon_i^{k+1} = \begin{cases} \min\left\{2\varepsilon_i^k, 10^4\right\} & \text{if } \chi_i(x_g^k, z_g^k) > 0.95 \cdot \chi_i(x_g^{k-1}, z_g^{k-1}) \\ \max\left\{\frac{1}{2}\varepsilon_i^k, 10^{-4}\right\} & \text{if } \chi_i(x_g^k, z_g^k) < 0.90 \cdot \chi_i(x_g^{k-1}, z_g^{k-1}) \\ \varepsilon_i^k & \text{otherwise.} \end{cases} \tag{12}$$

  In all the other iterations, $\varepsilon_i^{k+1} = \varepsilon_i^k$, $i = 1, \ldots, 7$.

The above argument implies that the internal loop stop condition in Algorithm 1 is $(k \bmod 5) = 0$. The choice of the coefficients used in (11) and (12) is motivated by efficiency reasons, and is obtained after a very coarse initial tuning over our portfolio selection problems.

Roughly speaking, in (11), penalty parameter ε_0^{k+1} is increased in $P(x, z; \varepsilon^{k+1})$ to favor optimality of solutions, possibly at the expenses of their feasibility, when the risk function value $\rho_{a,p}(r_g^k)$ increases. Following a similar argument, ε_0^{k+1} is decreased in order to increase feasibility when $\rho_{a,p}(r_g^k)$ decreases. As regards (12), penalty parameter ε_i^{k+1} is increased to favor feasibility of solutions possibly at the expenses of their optimality, when the violation $\chi_i(x_g^k, z_g^k)$ of the i-th constraint significantly increases with respect to $\chi_i(x_g^{k-1}, z_g^{k-1})$. Conversely, with an opposite rationale, the parameter ε_i^{k+1} is decreased in case we observe a relevant improvement of feasibility with respect to the i-th constraint i.e., $\chi_i(x_g^k, z_g^k) \ll \chi_i(x_g^{k-1}, z_g^{k-1})$.

4.2 Splitting and refining particles' positions

We observe that from (10) $P(x, z; \varepsilon^k)$ is convex with respect to the subvector x, as in (5) both $\rho_{a,p}(r)$ and the constrains functions are convex in $\mathbb{R}^N$ with respect to x. We try to take advantage of this fact in our hybrid metaheuristic in order to rapidly identify a (sub)-optimal value of the x component of the problem solution. In particular, at any iteration in the internal loop of PSO-D, we split each particle position in its components x_j^k and z_j^k and update them separately. For any particle j the subvector $z_j^{k+1} = z_j(x^k)$ is updated with the following procedure

$$(z_j^{k+1})_i = \begin{cases} 0 & \text{if } (x_j^k)_i \in (-\infty, d) \cup (u, \infty), \\ 1 & \text{otherwise.} \end{cases} \qquad j = 1, \ldots, P,\ i = 1, \ldots, N.$$

Then, we keep z_j^{k+1} constant and minimize $P(x, z_j^{k+1}; \varepsilon^k)$ only with respect to x, obtaining $\hat{x}_j^{k+1}$. Finally, $\hat{x}_j^{k+1}$ is further refined to obtain x_j^{k+1} as

$$(x_j^{k+1})_i = \frac{(\hat{x}_j^{k+1})_i (z_j^{k+1})_i}{\sum_{i=1}^N (\hat{x}_j^{k+1})_i (z_j^{k+1})_i}, \qquad j = 1, \ldots, P,\ i = 1, \ldots, N.$$

The above splitting and refining steps ensure that at the end of the internal loop of PSO-D, each vector (x_j^{k+1}, z_j^{k+1}) satisfies (5c), (5e) and (5f). In our numerical experience, we observed that the refinement of the subvector x has also another positive effect, since the value of the fitness function $\rho_{a,p}(r)$ at x_j^k is typically smaller than at $\hat{x}_j^k$. On the other hand, constraints (5b) and (5d) might be sometimes violated at (x_j^k, z_j^k). However, they are typically fulfilled in a neighborhood of the final solution point.

We complete this section by observing that, splitting the vector of unknowns in the two subvectors x and z, which are separately updated, may also be motivated from the perspective of the *Schwarz Alternating Method* (SAM), introduced by Gander (2008). The SAM method, which was originally conceived to speed up the solution of a differential equation on the union of a finite number of domains, can be extended to accelerate the solution of linear and nonlinear systems of equations. It is essentially based on splitting the set of variables into subsets. Then, the problem is repeatedly solved only on the resulting subsets of the unknowns, so that the overall problem is never fully solved with respect to all the variables.

5 Numerical experiences

In this section we report our experimental analysis. Specifically, we implemented:

1. our hybrid metaheuristic PSO-D as described by the Algorithm 1;
2. a PSO metaheuristic, hereafter referred to as PSO-S (*S* stands for *static*), in which penalty parameters vector ε is a-priori fixed for all the iterations;
3. a PSO metaheuristic, hereafter referred to as PSO-R (*R* stands for *REVAC*), with a REVAC parameter tuning approach (Nannen and Eiben 2007b; Montero et al. 2014), in which penalty parameters vector $\bar{\varepsilon}$ is first computed in a *presolve procedure* using REVAC. Then, we set $\varepsilon^k = \bar{\varepsilon}$, for any $k \geq 1$, when minimizing $P(x, z; \varepsilon^k)$ in (10);
4. a PSO metaheuristic, hereafter referred to as PSO-I (*I* stands for *irace*), with an *irace* parameter tuning approach (López-Ibáñez et al. 2016) in place of REVAC one.

Finally, we also treated (5) as a fully nonlinear mixed-integer problem, that we solved through a standard exact solver based on a Branch-and-Bound scheme (hereafter referred to as ES). We used the results obtained by exactly solving the mixed integer formulation to obtain reference values for the results provided by PSO-D, PSO-S, PSO-R and PSO-I. In this regard, note that since our portfolio selection problem is NP-hard, the ES approach may require a prohibitive amount of time for computation when the number of assets increases. This fact may obviously discourage practitioners from using it. Details of both the mixed-integer formulation and the relative solver adopted are reported in the Appendix.

As for the numerical instances, we considered assets belonging to stock-exchange indexes, in which daily close prices over a time horizon T are converted in daily returns by using the formula $r_{i,t} = \log\left(\frac{S_{i,t}}{S_{i,t-1}}\right)$, where $S_{i,t}$ represents the price of asset i at time t, and $r_{i,t}$ represents the return of asset i at time t. Then, in accordance with (Corazza et al. 2013; Chen and Wang 2008), we approximate the expected values that appear in the objective function (5a) with the following sample means:

$$\|(r - E[r])^+\|_1 = \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i=1}^{N}(r_{i,t} - \hat{r}_i)x_i\right)^+$$

$$\|(r - E[r])^-\|_p = \frac{1}{T^{\frac{1}{p}}} \left(\sum_{t=1}^{T} \left[\left(\sum_{i=1}^{N} (r_{i,t} - \hat{r}_i) x_i \right)^- \right]^p \right)^{\frac{1}{p}},$$

so that

$$\rho_{a,p}(r) = \frac{a}{T} \sum_{t=1}^{T} \left(\sum_{i=1}^{N} (r_{i,t} - \hat{r}_i) x_i \right)^+ + \frac{1-a}{T^{\frac{1}{p}}} \left(\sum_{t=1}^{T} \left[\left(\sum_{i=1}^{N} (r_{i,t} - \hat{r}_i) x_i \right)^- \right]^p \right)^{\frac{1}{p}} - \sum_{i=1}^{N} \hat{r}_i x_i. \tag{13}$$

Before passing to the detailed presentation of our numerical experiences, it is noteworthy to highlight that in the previous study (Corazza et al. 2012), PSO-S was applied to approximately solve the l_1-penalty problem (10), and its performances have been compared with those from the application of standard Genetic Algorithms (GAs). Note that GAs can be considered as an unquestioned benchmark in the field of evolutionary population-based metaheuristics. The results of this comparison have shown that the two metaheuristics are more or less equivalent, both in terms of fitness function values and of risk measure values, but the average computational time required by GAs is about one order magnitude greater than that required by PSO-S. This motivated our choice for a PSO-based approach, in the current paper.

5.1 Basics on REVAC

REVAC (Nannen and Eiben 2007b; Montero et al. 2014) is an Estimation of Distribution Algorithm used to tune a-priori the value of a vector of parameters of an algorithm. It relies on information theory to measure parameter relevance. Roughly speaking, REVAC considers a value distribution over the parameter space, i.e., the set of the possible values for each parameter. Specifically, REVAC assigns high probabilities to values leading to a good compromise between the algorithm performance and the algorithm complexity. Complexity is expressed in term of Shannon entropy.

REVAC is an iterative algorithm: it initially creates a uniform distribution over the parameters space, then this distribution is iteratively refined (*smoothed* in REVAC jargon). This is done by an evolutionary process that starts from an initial parameter vector population. Then, it generates new parameter vectors by choosing the best subset of vectors with respect to expected performance, in order to replace the eldest individuals in the population (Eiben and Smith 2003). In our case REVAC estimates the expected performance associated to a vector, by running PSO on small/medium size instances of the portfolio selection problem randomly generated.

The smoothing feature is assured by an operator that defines a mutation interval for each parameter. At each iteration, it sorts the current population parameter values and defines a new distribution by deleting a given number of extreme values. Then, it uses this new distribution to draw the next population parameter values randomly. The Shannon entropy is supposed to decrease over iterations, and we can use the information gathered to infer information on the parameters. Namely, parameters that show a great decrease of entropy are likely the most sensitive to their values, hence they are the most promising for parameter value choices (Nannen and Eiben 2007a).

We first ran REVAC to understand the relative relevance of parameters $\omega_1, \ldots, \omega_7$, being $\omega_i = \varepsilon_i / \varepsilon_0$ and $\varepsilon = (\varepsilon_0, \varepsilon_1, \cdots, \varepsilon_7)^T$ is the penalty parameters vector in (10). We identified

the 3 most sensitive parameters by selecting the 3 parameters whose entropy most decreases over the REVAC execution. Hence, we used the outcome of this run to set the values of the remaining parameters, by selecting their values in a pre-defined neighbourhood of the median of their resulting performance distributions. Then, we re-ran REVAC (using these latter parameters values) and at the end of the run we set the values of the remaining parameters, also by selecting their values in a pre-defined neighbourhood of the median of their resulting performance distributions.

As regards the code of REVAC, the MATLAB implementation by Volker Nannen we adopted is publicly available at http://www.complexity-research.net/revac.

5.2 Basics on irace

Irace (López-Ibáñez et al. 2016) is an Iterated RACing procedure which extends the Iterated F-race algorithm (Balaprakash et al. 2007; Bartz-Beielstein et al. 2010). Similarly to REVAC, it is used to automatically configure optimization algorithms, typically metaheuristics. It finds the appropriate values of a solution algorithm parameters given a set of tuning instances of an optimization problem. *Irace* is based on the repeated application of the racing technique introduced in Maron and Moore (1997), which

- tests a set of parallel configurations,
- quickly discards the ones that are clearly inferior,
- concentrates on differentiating among the better ones.

At each iteration *irace* selects the configurations to be discarded, by applying statistical tests such as the Friedman's non-parametric two-way analysis of variance by ranks, its extensions or the paired sample t-test. In addition, *irace* adopts an adaptive capping mechanism, which reduces the time wasted in the evaluation of poorly performing configurations, by bounding the maximum running time permitted for each such evaluation (Cáceres et al. 2017).

As for REVAC, we ran *irace* to determine the values of the parameters ω and ε (see Sect. 5.1). We used 18 tuning instances and we set a budget of 1000 calls to *targetRunner* as the maximum number of iterations.

As regards the code of *irace*, we used the R implementation which is publicly available at https://www.r-project.org/package=irace. In particular, we edited the appropriate R script to execute MATLAB on our PSO algorithm over the tuning instances.

5.3 PSO parameter settings and data

Similarly to what was done for the assessment of the penalty parameters (see Sect. 4.1), here we consider two different approaches for the PSO parameter setting, too.

In the first one, we used the same setting for PSO-D, PSO-S, PSO-R and PSO-I. Namely, we set $c_j^k = c_g^k = 1.49618$, $w^k = 0.7298$ and $\chi^k = 1$ (see also Eberhart and Shi 2000; Serani et al. 2016). Furthermore, we used the same random values for the initial positions and velocities of the particles of PSO-D, PSO-S, PSO-R and PSO-I. By doing so, we allowed full comparability of the results coming from the various setting strategies of the penalty parameters.

In the second approach, for our PSO-D we set c_j^k, c_g^k and w^k by adopting some dynamic updating rules, while for PSO-R and PSO-I we determined these parameters by applying the REVAC-based tuning procedure and the *irace*-based one, respectively. Regarding the dynamic updating, in order to overcome some drawbacks of the standard PSO, we considered

rules that have shown to be both methodologically founded (see, e.g. Ratnaweera et al. 2004b) and operationally effective when applied to portfolio selection problems (see, e.g. Guang-Feng et al. 2012). In particular, for c_j^k and c_g^k we respectively used

$$c_j^k = c_{j,\max} + \left(c_{j,\min} - c_{j,\max}\right)\frac{k}{K} \quad \text{and} \quad c_g^k = c_{g,\min} + \left(c_{g,\max} - c_{g,\min}\right)\frac{k}{K},$$

where $c_{j,\min} = c_{g,\min} = 0.5$ and $c_{j,\max} = c_{g,\max} = 2.5$ according to the prevailing literature, and K indicates the maximum number of iterations. Notice that c_j^k is linearly decreasing while c_g^k is linearly increasing. Moreover, the last choice ensures a remarkable global search at the beginning of the process and a deeper exploitation at the end of the process, with respect to the standard PSO. At the same time, for w^k we used

$$w^k = w_{\min} + (w_{\max} - w_{\min})\frac{K-k}{K} \quad \text{and} \quad \chi^k = 1,$$

where $w_{\min} = 0.4$ and $w_{\max} = 0.9$, again according to the prevailing literature. Note that w^k is linearly decreasing and that its dynamics mainly steers exploration at the outset of the iterative process, while privileging exploitation in the end compels the particles mainly to explore at the beginning of the search and mainly to exploit towards the end of the iterations.

As for the setting of the parameters of the risk measure (5a), it is standard. We used $a = 0.5$ and $p = 2$ as often considered in literature (see, for instance, Corazza et al. 2013; Chen and Wang 2008). Conversely, we tried different settings of the parameters of the constraints (5d) to test our hybrid metaheuristic in various stressing configurations. Specifically, we considered three different pairs of cardinality constraints: $(K_d, K_u) = (5, 9)$ for small portfolios to select within a small cardinality range, $(K_d, K_u) = (30, 34)$ for large portfolios to select within a small cardinality range, and $(K_d, K_u) = (5, 34)$ for small-to-large portfolios to select within a large cardinality range. For all these three (K_d, K_u) configurations, we used the same values of the minimum and maximum percentages of capital to invest in each asset, namely $d = \frac{1}{34}$ and $u = \frac{1}{5}$ in (5e). In other words, we allowed the same capital investment for each asset in all the investigated configurations. Finally, for all the three (K_d, K_u) configurations, we have considered three values of the minimum desired expected return of the portfolio r_e close to the maximum expected return of portfolio achievable in that scenario.

Regarding the financial data, we considered the daily closing prices of 38 of the 40 assets composing the Italian stock index FTSE MIB, from January 15, 2016 to March 15, 2018, for a total of 564 prices for any asset. The exclusion of two assets is due to the fact that their listing on the Italian stock exchange started after January 15, 2016.

Lastly, as for the implementations of PSO-D, PSO-S, PSO-R and PSO-I, we set $P = 2 \cdot 38 = 76$, i.e. we choose the number of particles as twice the number of variables, a standard setting in several PSO-based methods (see, for instance Serani et al. 2016).

5.4 Results

In this subsection, we show that PSO-D is competitive with all PSO-S, PSO-R and PSO-I. Furthermore, we also perform a comparison between PSO-D, PSO-R and PSO-I and ES, showing that PSO-D is competitive in terms of CPU-time, at a modest expenses of the accuracy of the provided solutions.

We ran the metaheuristics PSO-D, PSO-S, PSO-R and PSO-I 25 times, performing 100 iterations each, for any configuration of the portfolio selection problem described in Sect. 3. We chose both the number of runs and the number of iterations unusually small, to show

the effectiveness of our hybrid metaheuristic PSO-D even in presence of few runs and a few iterations.

Hereinafter, with reference to the portfolios detected by the metaheuristics PSO-D, PSO-S, PSO-R and PSO-I, we use the terminology which follows. By *optimal portfolio* (OP) we mean the best portfolio in terms of fitness detected by a swarm at the end of a run after 100 iterations. By *global optimal portfolio* (GOP) we mean the best portfolio in terms of fitness detected during all the 25 runs by the various swarms facing the same configurations of the portfolio selection problem. In other words, GOP is the best portfolio, in terms of fitness, among the 25 detected OPs.

With reference to the first PSO parameter setting approach (i.e., $c_j^k = c_g^k = 1.49618$, $w_k = 0.7298$ and $\chi^k = 1$), in Table 1 we report the results of PSO-D and we compare them with those of PSO-S, PSO-R and PSO-I, and in Table 2 we compare the results of PSO-D, PSO-R and PSO-I with the ones by the exact solver ES. Analogously for Tables 3 and 4 , which report the results from adopting the second PSO parameter setting approach. Lastly, as an example, in Fig. 1 we also graphically plot some results from the numerical experiments.

As we describe in the following, the GOPs provided by our hybrid metaheuristic PSO-D are generally not worse in terms of quality of the solutions than those provided by PSO-S, PSO-R and PSO-I, and are always better in terms of the computational time required to calculate them.

(1) *PSO-D versus PSO-R and PSO-I* Tables 1 and 3 compare the results of PSO-D, PSO-R and PSO-I when applied to the configurations of portfolio selection problem introduced in Sect. 5.3. In particular, column 3 (MH) indicates the metaheuristic from which the results come. Columns 4 (P_{PRE}) and 5 (P) respectively report the values of the fitness before and after the final refinement is applied. Analogously for columns 6 (ρ_{PRE}) and 7 (ρ), but with respect to the value of the risk measure of the portfolio return, and for columns 8 (r_{PRE}) and 9 (r), but with respect to the achieved minimum expected return. Column 10 (#) provides the number of assets selected in the GOP. Column 11 ($\%\ <_P$) gives the percentages of OPs generated by PSO-D, PSO-R and PSO-I that respectively have fitness values lower than the ones associated to the OPs generated by PSO-S. Finally, column 12 ($\%F$) reports the percentages of OPs which are feasible. We highlight that for comparative purposes we also applied the final refinement to PSO-R and to PSO-I.

Table 1 suggests that, jointly considering the solution quality and the required computational time, the results of PSO-D are definitely better than those of PSO-R and PSO-I. Table 3 suggests the same, although a slight worsening of the quality of all the solutions is detectable. We will return to this aspect shortly.

Now, consider the effectiveness of the refinement procedure in PSO-D. In general, the values reported in columns 4 and 5 of Tables 1 and 3 point out a dramatic decrease of the fitness due to the refinement of the solutions. The refinement tends to reduce/cancel the infeasibility of the solution. Moreover, the values of the risk measure after refinement (column 7 of Tables 1 and 3) also improve with respect to the values of the risk measure before refinement (column 6 of Tables 1 and 3), although not as evidently as for the fitness.

Then, consider the feasibility of the solution provided by PSO-D. Constraints (5b)–(5f) are often violated (column 12 of Tables 1 and 3). Anyway, their violations are generally very small, so that to large extent infeasibility is not really a concern for OPs using PSO-D.

Finally, consider the comparison between PSO-D, PSO-R and PSO-I from Tables 1 and 3 . The performances of PSO-D, PSO-R and PSO-I in terms of fitness of the solution, of risk measure of the portfolio return, of the achieved minimum expected return of the portfolio and of the number of selected assets (columns 5, 7, 9 and 10 of Tables 1 and 3 , respectively) are

Table 1 Results of metaheuristics (MH) PSO-D, PSO-S, PSO-R and PSO-I, when applied to the first PSO parameter setting approach for different configurations of r_e and of (K_d, K_u), for the portfolio selection problem introduced in Sect. 5.3

(K_d, K_u)	r_e	MH	P_{PRE}	P	ρ_{PRE}	ρ	r_{PRE}	r	#	$\%<_P$	$\%F$
(5, 9)	0.00140	PSO-D	16148.7451811	0.0061137	0.0048897	0.0061137	0.0014046	0.0014028	6	76	4
		PSO-R	0.9478752	0.0059689	0.0070629	0.0059254	0.0013565	0.0013565	6	88	0
		PSO-I	2526.8560136	0.0062079	0.0071907	0.0060022	0.0011944	0.0011944	8	96	0
	0.00141	PSO-D	19485.2080029	0.0060230	0.0107327	0.0060171	0.0013800	0.0013792	6	88	0
		PSO-R	0.9477851	0.0059451	0.0043986	0.0059293	0.0013942	0.0013942	6	100	0
		PSO-I	0.8888522	0.0061466	0.0090431	0.0058624	0.0011258	0.0011258	8	96	0
	0.00142	PSO-D	465.4482294	0.0060846	0.0058445	0.0060188	0.0013731	0.0013733	6	76	0
		PSO-R	0.9475832	0.0060338	0.0126773	0.0059429	0.0013555	0.0013556	6	88	0
		PSO-I	193.3920238	0.0057247	0.0180303	0.0055247	0.0012200	0.0012200	8	92	0
(30, 34)	0.00040	PSO-D	104.4374480	1.2827674	0.0219675	0.0063189	0.0012234	0.0003771	29	68	0
		PSO-R	89685.0228688	7.0859903	0.0296697	0.0058848	−0.0001248	0.0003509	23	76	0
		PSO-I	5437.1586080	7.0591821	0.0202261	0.0062264	0.0007407	0.0003772	23	60	0
	0.00045	PSO-D	262869.6284558	4.0395871	0.0131589	0.0058749	0.0016571	0.0004782	22	80	0
		PSO-R	271278.6943815	12.0283537	0.0155525	0.0063526	−0.0033779	0.0001380	18	60	0
		PSO-I	9.6880220	4.1269271	0.0203488	0.0065186	0.0007593	0.0003315	26	84	0
	0.00050	PSO-D	76848.5294234	0.2171283	0.0701796	0.0059629	0.0008654	0.0003767	24	56	0
		PSO-R	246488.2926799	4.0787165	0.0127564	0.0062517	0.0001686	0.0003084	26	56	0
		PSO-I	220878.0957535	8.0726378	0.0498235	0.0062545	0.0022996	0.0003789	22	60	0
(5, 34)	0.00120	PSO-D	11621.4295240	0.0061270	0.0076818	0.0061270	0.0012994	0.0012994	6	76	48
		PSO-R	19.9300988	0.0060595	0.0066929	0.0060595	0.0013841	0.0013841	6	100	96
		PSO-I	2.5512554	0.0066348	0.0231238	0.0053300	0.0008587	0.0004179	10	96	0
	0.00130	PSO-D	4478.5775150	0.0061451	0.0052209	0.0061451	0.0013114	0.0013115	6	84	28
		PSO-R	2.2200054	0.0060745	0.0072771	0.0060745	0.0013247	0.0013247	6	96	48
		PSO-I	2792.7444536	0.0168930	0.0184170	0.0076884	0.0001136	0.0001065	14	92	0
	0.00140	PSO-D	12488.0754322	0.0061128	0.0099441	0.0061027	0.0013736	0.0013737	6	88	0
		PSO-R	0.9473530	0.0060621	0.0067296	0.0059884	0.0013263	0.0013263	6	100	0
		PSO-I	5168.5823313	0.0089302	0.0757703	0.0069974	−0.0005277	0.0001319	12	92	0

Table 2 Results of metaheuristic (MH) PSO-D, PSO-R, PSO-I and of the exact solver ES, when applied to the first PSO parameter setting approach for different configurations of r_e and of (K_d, K_u), for the portfolio selection problem introduced in Sect. 5.3

(K_d, K_u)	r_e	ρ_{ES}	r_{ES}	$\#_{ES}$	MH	$\%_\rho$	$\%_r$	$\%_\#$	$\%_{In}$	$\%_{Out}$
(5, 9)	0.00140	0.0059199	0.00140	6	PSO-D	3.27	0.20	0.00	100.00	0.00
					PSO-R	0.09	−3.10	0.00	100.00	0.00
					PSO-I	1.39	−14.69	33.33	100.00	25.00
	0.00141	0.0059858	0.00141	6	PSO-D	0.52	−2.19	0.00	100.00	0.00
					PSO-R	−0.94	−1.12	0.00	100.00	0.00
					PSO-I	−2.06	−20.15	33.33	100.00	25.00
	0.00142	0.0060699	0.00142	6	PSO-D	−0.84	−3.29	0.00	100.00	0.00
					PSO-R	−2.09	−4.54	0.00	100.00	0.00
					PSO-I	−8.98	−14.08	33.33	100.00	25.00
(30, 34)	0.00040	0.0055508	0.00040	30	PSO-D	13.84	−5.73	−3.33	90.00	6.90
					PSO-R	6.02	−12.27	−23.33	66.67	13.04
					PSO-I	12.17	−5.70	−23.33	70.00	8.70
	0.00045	0.0056000	0.00045	30	PSO-D	4.91	6.27	−26.67	66.67	9.09
					PSO-R	13.44	−69.30	−40.00	60.00	0.00
					PSO-I	16.40	−26.32	−13.33	76.67	11.54
	0.00050	0.0057148	0.00050	30	PSO-D	4.34	−24.67	−20.00	73.33	8.33
					PSO-R	13.44	−69.30	−40.00	60.00	0.00
					PSO-I	9.44	−24.23	−26.67	73.33	0.00
(5, 34)	0.00120	0.0049390	0.00120	8	PSO-D	24.05	8.28	−25.00	62.50	16.67
					PSO-R	22.69	15.34	−25.00	62.50	16.67
					PSO-I	7.92	−65.17	25.00	25.00	80.00
	0.00130	0.0053548	0.00130	8	PSO-D	14.76	0.88	−25.00	75.00	0.00
					PSO-R	13.44	1.90	−25.00	75.00	0.00
					PSO-I	43.58	−91.81	75.00	25.00	85.71
	0.00140	0.0059199	0.00140	6	PSO-D	3.09	−1.88	0.00	100.00	0.00
					PSO-R	1.16	−5.26	0.00	100.00	0.00
					PSO-I	18.20	−90.58	100.00	0.00	100.00

Table 3 Results of metaheuristics (MH) PSO-D, PSO-S, PSO-R and PSO-I, when applied to the second PSO parameter setting approach for different configurations of r_e and of (K_d, K_u), for the portfolio selection problem introduced in Sect. 5.3

(K_d, K_u)	r_e	MH	P_{PRE}	P	ρ_{PRE}	ρ	r_{PRE}	r	#	$\%<_P$	$\%F$
(5, 9)	0.00140	PSO-D	1366.0010140	0.4192102	0.0114570	0.0055857	0.0008460	0.0002832	9	88	0
		PSO-R	66.530667	0.0429682	0.0161805	0.0062613	0.0011267	0.0011457	6	76	0
		PSO-I	522.5437336	0.0062180	0.0062180	0.0060507	0.0012295	0.0012326	7	92	0
	0.00141	PSO-D	276.0224043	0.1447927	0.0206151	0.0064724	0.0005068	0.0002896	9	94	0
		PSO-R	30411.3497649	0.0072167	0.0130021	0.0067590	0.0010865	0.0010857	8	72	0
		PSO-I	0.8893458	0.0063745	0.0153320	0.0061606	0.0011899	0.0011960	8	92	0
	0.00142	PSO-D	1224.0868847	0.4338280	0.0207993	0.0062869	0.0003105	0.0002656	9	96	0
		PSO-R	7001.1838130	0.0191582	0.0059899	0.0064988	0.0011553	0.0011653	7	68	0
		PSO-I	0.8916808	0.0062043	0.0064772	0.0059368	0.0011561	0.0011525	8	96	0
(30, 34)	0.00040	PSO-D	5907.6993584	65.2561484	0.0317071	0.0067094	0.0001404	0.0000608	30	100	0
		PSO-R	17265.1362496	1976.4244266	0.0259374	0.0066724	−0.0003132	0.0003881	14	60	0
		PSO-I	325206.8759770	1080.0557450	0.0199062	0.0054213	0.0043717	0.0006999	4	40	0
	0.00045	PSO-D	3869.4693098	63.4463109	0.0223458	0.0073250	0.0000708	0.0001338	30	84	0
		PSO-R	96700.5403025	13.7879717	0.0302456	0.0066495	0.0015120	0.0003278	21	60	0
		PSO-I	337665.5585815	2722.6262579	0.0210941	0.0115706	−0.0007705	−0.0000338	8	56	0
	0.00050	PSO-D	143950.6403430	91.7149916	0.0194019	0.0065507	0.0001855	0.0001188	30	92	0
		PSO-R	92418.8183684	11.0367018	0.0221554	0.0061100	0.0007972	0.0004693	19	60	0
		PSO-I	40442.6223619	24.3928319	0.0336603	0.0057007	0.0026773	0.0001032	14	64	0
(5, 34)	0.00120	PSO-D	8292.4391150	3.1121015	0.0289370	0.0068554	0.0003657	0.0002684	13	96	0
		PSO-R	23725.0213649	0.0126412	0.0792188	0.0080114	−0.0012655	0.0001875	9	68	0
		PSO-I	95495.9291026	0.0075776	0.0299612	0.0066967	−0.0009127	0.0003191	12	72	0
	0.00130	PSO-D	10441.3096048	1.1169603	0.0268656	0.0073671	0.0013930	0.0003014	16	96	0
		PSO-R	68933.3168542	0.0105371	0.0251846	0.0056465	−0.0012272	0.0002304	9	72	0
		PSO-I	588.1235878	0.0081574	0.0215704	0.0063752	0.0009105	0.0001177	12	68	0
	0.00140	PSO-D	7434.3913612	1.8585282	0.0271137	0.0070536	0.0003385	0.0002529	19	100	0
		PSO-R	471.7392033	0.0071050	0.0116356	0.0060078	−0.0000996	0.0003028	10	76	0
		PSO-I	7.6839161	0.0064787	0.0802989	0.0053113	−0.0008392	0.0002327	10	80	0

Table 4 Results of metaheuristic (MH) PSO-D, PSO-R, PSO-I and of the exact solver ES, when applied to the second PSO parameter setting approach for different configurations of r_e and of (K_d, K_u), for the portfolio selection problem introduced in Sect. 5.3

(K_d, K_u)	r_e	ρ_{ES}	r_{ES}	$\#_{ES}$	MH	$\%_\rho$	$\%_r$	$\%_\#$	$\%_{In}$	$\%_{Out}$
(5, 9)	0.00140	0.0059199	0.00140	6	PSO-D	−5.65	−79.77	50.00	33.33	77.78
					PSO-R	5.77	−18.17	0.00	66.67	33.33
					PSO-I	2.21	−11.95	16.67	100.00	14.29
	0.00141	0.0059858	0.00141	6	PSO-D	8.13	−79.46	50.00	33.33	77.78
					PSO-R	12.91	−23.00	33.33	100.00	25.00
					PSO-I	2.92	−15.17	33.33	100.00	25.00
	0.00142	0.0060699	0.00142	6	PSO-D	3.57	−81.29	50.00	33.33	77.78
					PSO-R	7.07	−17.93	16.67	100.00	14.29
					PSO-I	−2.19	−18.84	33.33	100.00	25.00
(30, 34)	0.00040	0.0055508	0.00040	30	PSO-D	20.87	−84.80	0.00	83.33	16.67
					PSO-R	20.21	−2.97	−53.33	40.00	14.29
					PSO-I	−2.33	74.97	−86.66	13.33	0.00
	0.00045	0.0056000	0.00045	30	PSO-D	30.80	−70.28	0.00	80.00	20.00
					PSO-R	18.74	−27.15	−30.00	66.67	4.76
					PSO-I	106.62	−107.52	−73.33	26.67	0.00
	0.00050	0.0057148	0.00050	30	PSO-D	14.63	−76.23	0.00	86.67	13.33
					PSO-R	6.91	−6.13	−36.66	56.67	10.53
					PSO-I	−0.24	−79.35	−53.33	43.33	7.14
(5, 34)	0.00120	0.0049390	0.00120	8	PSO-D	38.80	−77.63	62.50	50.00	69.23
					PSO-R	62.21	−84.37	12.50	25.00	77.78
					PSO-I	35.59	−73.41	50.00	62.50	58.33
	0.00130	0.0053548	0.00130	8	PSO-D	37.57	−76.82	100.00	62.50	68.75
					PSO-R	5.45	−82.27	12.50	37.50	66.67
					PSO-I	19.05	−90.94	50.00	12.50	91.67
	0.00140	0.0059199	0.00140	6	PSO-D	19.15	−81.93	216.67	50.00	84.21
					PSO-R	1.48	−78.37	66.67	33.33	80.00
					PSO-I	−10.28	−83.38	66.67	16.67	90.00

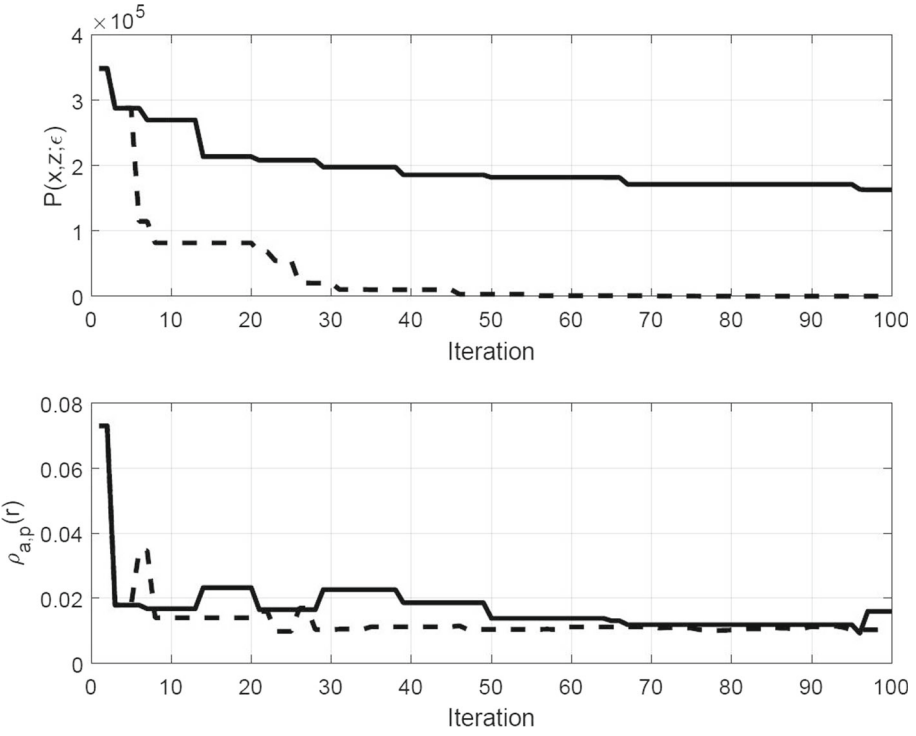

Fig. 1 In the upper panel, the average values over the iterations of the fitness functions of PSO-D and PSO-S (dashed and continuous line, respectively). In the lower panel, the average values over the iterations of the risk measures. These values refer to the second run (out of 25) of PSO-D and PSO-S, associated to the first PSO parameter setting approach and to the configuration with parameter setting $(K_d, K_u) = (30, 34)$ and $r_e = 0.00045$

generally similar. Nevertheless, PSO-D needs on average less then 4% of the computational time requested by PSO-R or PSO-I.

Now consider as an example Fig. 1. It is structured in two panels. The upper one presents the average value of the fitness function at particles position, for PSO-D and PSO-S, as a function of the number of iterations (dashed and continuous line, respectively). The trends of these values are representative for the great majority of the considered configurations of the portfolio selection problem. PSO-D induces a faster decrease of the fitness function since the early iterations. In addition, Column 11 of Table 1 shows that 80.89% of the OPs generated by PSO-D have a final fitness which is lower than the fitness of the corresponding OPs generated by PSO-S. Analogous arguments apply to the lower panel of Fig. 1, which presents the values of the risk measures returned by PSO-D and PSO-S.

(2) PSO-D, PSO-R, and PSO-I vs. ES. Tables 2 and 4 compare the results returned by PSO-D, PSO-R and PSO-I with the one generated by the exact solver ES, when applied to the configurations of portfolio selection problem introduced in Sect. 5.3. In particular, columns 3 (ρ_{ES}), 4 (r_{ES}) and 5 ($\#_{ES}$) respectively report the values of the risk measure, of the achieved minimum expected return and the number of selected assets provided by ES. Columns 7 ($\%_\rho$), 8 ($\%_r$) and 9 ($\%_\#$) respectively report the percentage values of the differences of the risk measure, of the achieved minimum expected return and of the number of selected assets provided by PSO-D, PSO-R and PSO-I, with respect to the corresponding values of ES.

Column 10 ($\%_{In}$) gives the percentages of assets selected by ES which have been selected also in the portfolio generated by PSO-D, PSO-R and PSO-I (the desired value is possibly 100%) and column 11 ($\%_{Out}$) reports the percentages of assets selected in the portfolio generated by PSO-D, PSO-R and PSO-I, which have not been selected also by ES (the desired value is possibly 0%). For the sake of completeness, we strongly remark that the formulation (5) does not admit in general a unique solution.

The GOPs generated by our hybrid metaheuristic PSO-D are obviously sub-optimal when compared to those selected by the exact solver ES. However they are to large extent reasonably acceptable in practice. Indeed, with particular reference to the first PSO parameter setting approach, with the exception of a few cases, the values of the indicators $\%_\rho$ and $\%_r$ (see the results in columns 7 and 8 of Table 2) are close to 0. Moreover, several portfolios generated by PSO-D are much "similar" to the ES ones. Indeed, the percentages of assets selected in both the ES portfolios and in the PSO-D ones is generally high (see the results in column 10 of Table 2). Conversely, the percentage of assets selected in the PSO-S portfolios which were not selected in the ES ones is generally low. Similar results also hold for the PSO-R and the PSO-I portfolios.

Regarding the second PSO parameter setting approach, Table 4 confirms these results but in a weaker way, as already pointed out for those presented in Table 3. Indeed, all the considered indicators (i.e., $\%_\rho$, $\%_r$, $\%_\#$, $\%_{In}$ and $\%_{Out}$) are worse than the corresponding ones reported in Table 2. This widespread and meaningful numerical evidence indicates that ad hoc settings of the PSO parameters generally worsen the performance of our exact penalty-based approach, at least in presence of few runs and a few iterations.

6 Conclusions and future work

In this paper, starting from the results in Corazza et al. (2013), Corazza et al. (2012), we have proposed a novel hybrid metaheuristic based on PSO with a dynamic penalty approach, for rapidly solving complex mathematical programming problems. We have applied our hybrid approach to an unconstrained reformulation of a realistic portfolio selection problem. We have performed a set of experiments, comparing our PSO scheme with both an exact method and with a REVAC-based / *irace*-based variants of PSO that resort to a static parameter tuning approach. Results show that our proposal compares favourably with the exact solver and with the REVAC / *irace* approach, but it requires much less computational time. This aspect is of great interest for practitioners, that could introduce our approach in their daily operations. Furthermore, the comparison between our PSO scheme (i.e., a *dynamic parameter control* approach) and a *parameter tuning* approach based on REVAC / *irace* is useful, in the ongoing discussion about parameter settings, and can be applied to other meta-heuristics to study their performances.

Although we obtained satisfactory numerical results, our research seems to offer further opportunities for possible improvements and extensions. In particular, combining PSO with a globally convergent method for derivative-free optimization may possibly provide a better quality of the solutions (see also Griffin and Kolda 2010). This latter fact deserves additional investigation, in order to further exploit the structure of portfolio selection problems (i.e. the convexity of problem (5) with respect to the subvector of unknowns x).

Funding Open access funding provided by Università Ca' Foscari Venezia within the CRUI-CARE Agreement.

Declaration

Conflict of interest Authors declare that they have no significant competing financial, professional, or personal interests that might have influenced the performance or presentation of the work described in this manuscript.

Appendix A: Portfolio selection problem: a possible mixed-integer formulation

In this appendix, we reformulate problem (5) as a nonlinear mixed-integer programming problem. This reformulation provides a model whose instances can be solved by standard exact algorithms as FilMINT (which is an iterative method based on a Branch-and-Cut framework), publicly available on NEOS server (http://www.neos-server.org/neos/). The main purpose of this section is to provide a formulation of our portfolio selection problem, whose exact solution can be possibly used as a reference in our numerical experience (see Tables 1 and 2).

In accordance with (Chen and Wang 2008), the resulting nonlinear mixed-integer programming problem with $2N + 2T$ unknowns is:

$$\min_{x,z,\beta,\gamma} \quad \frac{a}{T}\sum_{t=1}^{T}\beta_t + \frac{1-a}{T^{\frac{1}{p}}}\left(\sum_{t=1}^{T}\gamma_t^p\right)^{\frac{1}{p}} - \sum_{i=1}^{N}\hat{r}_i x_i \tag{A.1a}$$

$$\beta_t - \gamma_t = \sum_{i=1}^{N}(r_{i,t} - \hat{r}_i)x_i, \qquad t = 1, \ldots, T \tag{A.1b}$$

$$\sum_{i=1}^{N}\hat{r}_i x_i \geq r_e \tag{A.1c}$$

$$\sum_{i=1}^{N} x_i = 1 \tag{A.1d}$$

$$K_d \leq \sum_{i=1}^{N} z_i \leq K_u \tag{A.1e}$$

$$dz_i \leq x_i \leq uz_i, \qquad i = 1, \ldots, N \tag{A.1f}$$

$$z_i \in \{0, 1\}, \qquad i = 1, \ldots, N \tag{A.1g}$$

$$\beta_t, \gamma_t \geq 0, \qquad t = 1, \ldots, T. \tag{A.1h}$$

In the given formulation, conditions (A.1c)–(A.1g) are trivially equivalent to (5b)–(5f). In addition, we write the objective function (5a) as in (13) where we introduce the variables $\beta^T = (\beta_1, \ldots, \beta_T)$, $\gamma^T = (\gamma_1, \ldots, \gamma_T)$ as a standard trick to linearize the terms

$\left(\sum_{i=1}^{N}(r_{i,t}-\hat{r}_{i,t})x_i\right)^+$ and $\left(\sum_{i=1}^{N}(r_{i,t}-\hat{r}_{i,t})x_i\right)^-$. Indeed, conditions (A.1b) and (A.1h) imply that

$$\beta_t \geq \left(\sum_{i=1}^{N}(r_{i,t}-\hat{r}_{i,t})x_i\right)^+ \quad \text{and} \quad \gamma_t \geq \left(\sum_{i=1}^{N}(r_{i,t}-\hat{r}_{i,t})x_i\right)^-, \quad t=1,\ldots,T. \tag{A.2}$$

In particular, we can rewrite (A.1b) as $\beta_t = \sum_{i=1}^{N}(r_{i,t}-\hat{r}_i)x_i + \gamma_t$. Hence, as $\gamma_t \geq 0$, we obtain $\beta_t \geq \sum_{i=1}^{N}(r_{i,t}-\hat{r}_i)x_i$ and, as $\beta_t \geq 0$, $\beta_t \geq \max\{\sum_{i=1}^{N}(r_{i,t}-\hat{r}_i)x_i, 0\}$. We finally observe that the inequalities (A.2) hold as equalities in any minimum of problem (A.1). As an example consider any feasible solution $(\bar{x},\bar{z},\bar{\beta},\bar{\gamma})$, for which the value of $\sum_{i=1}^{N}(r_{i,t}-\hat{r}_i)\hat{x}_i$ is nonnegative and such that $\bar{v}_t > \sum_{i=1}^{N}(r_{i,t}-\hat{r}_i)\bar{x}_i$ and $\bar{w}_t > 0$. Since variables v and w have positive coefficients in the objective function (A.1a), and no other constraint different from (A.1b) and (A.1h) involves β and γ, then it is immediate to find feasible solution $(\bar{x},\bar{z},\tilde{\beta},\tilde{\gamma})$ that dominates $(\bar{x},\bar{z},\bar{\beta},\bar{\gamma})$. Just set $\tilde{\beta} = \bar{\beta}$, respectively $\tilde{\gamma} = \bar{\gamma}$, except for $\tilde{\beta}_t = \sum_{i=1}^{N}(r_{i,t}-\hat{r}_i)\bar{x}_i$, respectively for $\tilde{\gamma}_t = 0$.

As indicated in Sect. 5, we use formulation (A.1) to obtain the optimal values ρ^* of our portfolio selection problem instances, to be compared with the values obtained through PSO. Unfortunately, the optimal solution of problem (A.1) may be particularly cumbersome to compute by an exact solver, when a large size instance is considered. All the same, we can use the linear relaxation of (A.1) to easily obtain a lower bound of ρ^* and, hence, to be able to asses the PSO performances even for a large size instance.

Indeed, we can observe that, if we relax constraints (A.1g) with $0 \leq z_i \leq 1$ for all i, the objective function (A.1a) becomes a convex functional as it is sublinear. In addition, also the feasible set defined by constrains (A.1b)–(A.1h) becomes convex. Consequently, all the feasible stationary points of the linearly relaxed version of (A.1) are optimal and the value assumed by the objective function (A.1a) at these points provides a lower bound for ρ^*. Due to the convexity properties of the relaxed version of (A.1a), its stationary points are relatively easy to determine using standard nonlinear optimization algorithms.

References

Arora, S., Barak, B., Brunnermeier, M., & Ge, R. (2011). Computational complexity and information asymmetry in financial products. *Communication ACM*, *54*(5), 101–107.

Artzner, P., Delbaen, F., Eber, J.-M., & Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, *9*(3), 203–228.

Balaprakash, P., Birattari, M., Stützle, T. (2007). Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In T. Bartz-Beielstein, M. Blesa, C. Blum, B. Naujoks, A. Roli, G. Rudolph, M. Sampels (Eds.), *Hybrid metaheuristics* Vol. 4771 of *Lecture Notes in Computer Science* (pp. 108–122). Springer, Heidelberg.

Bartz-Beielstein, T., Lasarczyk, C., & Preuss, M. (2010). The sequential parameter optimization toolbox. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, & M. Preuss (Eds.), *Experimental Methods for the Analysis of optimization algorithms* (pp. 337–360). Berlin: Springer.

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2006). *Nonlinear Programming: Theory and Algorithms* (3rd ed.). London: Wiley.

Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Proceedings of the 24th international conference on neural information processing systems, NIPS'11* (pp. 2546–2554). Curran Associates Inc., Red Hook, NY.

Bertsekas, D. P. (2016). *Nonlinear Programming* (3rd ed.). Nashua: Athena Scientific.

Birattari, M., Yuan, Z., Balaprakash, P., & Stützle, T. (2010). F-race and iterated f-race: An overview. In *Experimental Methods for the Analysis of Optimization Algorithms* (pp. 311–336). Springer.

Bonyadi, M. R., & Michalewicz, Z. (2016). Analysis of stability, local convergence, and transformation sensitivity of a variant of particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, *20*(3), 370–385.

Cáceres, L. P., López-Ibáñez, M., Hoos, H., & Stützle, T. (2017). An experimental study of adaptive capping in Irace. In R. Battiti, D. E. Kvasov, & Y. D. Sergeyev (Eds.), *Learning and Intelligent Optimization* (pp. 235–250). Cham: Springer.

Campana, E. F., Fasano, G., & Pinto, A. (2010). Dynamic analysis for the selection of parameters and initial population, in particle swarm optimization. *Journal of Global Optimization*, *48*(3), 347–397.

Chen, W., & Zhang, W.-G. (2010). The admissible portfolio selection problem with transaction costs and an improved pso algorithm. *Physica A: Statistical Mechanics and its Applications*, *389*(10), 2070–2076.

Chen, Z., & Wang, Y. (2008). Two-sided coherent risk measures and their application in realistic portfolio optimization. *Journal of Banking and Finance*, *32*, 2667–2673.

Corazza, M., di Tollo, G., Fasano, G., & Pesenti, R. (2019). A PSO-based framework for nonsmooth portfolio selection problems. *Neural Advances in Processing Nonlinear Dynamic Signals, Smart Innovations, Systems and Technologies*, *102*, 265–275.

Clerc, M., & Kennedy, J. (2002). The particle swarm—explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*(1), 58–73.

Corazza, M., Fasano, G., & Gusso, R. (2012). Portfolio selection with an alternative measure of risk: Computational performances of particle swarm optimization and genetic algorithms. In C. Perna & M. Sibillo (Eds.), *Mathematical and Statistical Methods for Actuarial Sciences and Finance* (pp. 123–130). Berlin: Springer.

Corazza, M., Fasano, G., & Gusso, R. (2013). Particle swarm optimization with non-smooth penalty reformulation, for a complex portfolio selection problem. *Applied Mathematics and Computation*, *224*, 611–624.

Dai, Y., Liu, L., & Li, Y. (2011). An intelligent parameter selection method for particle swarm optimization algorithm. In *2011 Fourth international joint conference on computational sciences and optimization* (pp. 960–964).

Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of IEEE congress on evolutionary computation* (pp. 84–88).

Eberhart, R. C., & Shi, Y. (2001). Particle swarm optimization: Developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No.01TH8546)* (Vol. 1, pp. 81–86).

Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H. H., & Leyton-brown, K. (2013). Towards an empirical foundation for assessing Bayesian optimization of hyperparameters In *In NIPS workshop on Bayesian optimization in theory and practice*.

Eggensperger, K., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2015). Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proceedings of the twenty-Ninth AAAI conference on artificial intelligence* (pp. 1114–1120). AAAI Press.

Eiben, A. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *3*(2), 124–141.

Eiben, A. E., & Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Berlin: Springer.

Fisher, M. L. (1985). An applications oriented guide to Lagrangian relaxation. *Interfaces*, *15*(2), 10–21.

Gander, M. J. (2008). Schwarz methods over course of time. *Electronic Transactions on Numerical Analysis*, *31*, 228–255.

Griffin, J. D., & Kolda, T. G. (2010). Nonlinearly constrained optimization using heuristic penalty methods and asynchronous parallel generating set search. *Applied Mathematics Research Express*, *2010*(1), 36–62.

Guang-Feng, D., Woo-Tsong, L., & Chih-Chung, L. (2012). Markowitz-based portfolio selection with cardinality constraints using improved particle swarm optimization. *Expert Systems with Applications*, *4*(39), 4558–4566.

Harrison, K. R., Engelbrecht, A. P., & Ombuki-Berman, B. M. (2018). Self-adaptive particle swarm optimization: A review and analysis of convergence. *Swarm Intelligence*, *12*(3), 187–226.

Hong, W.-C. (2009). Chaotic particle swarm optimization algorithm in a support vector regression electric load forecasting model. *Energy Conversion and Management*, *50*(1), 105–117.

Hsieh, S.-T., Sun, T.-Y., Liu, C.-C., & Tsai, S.-J. (2009). Efficient population utilization strategy for particle swarm optimizer. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *39*(444–456), 05.

Huang, C., Li, Y., & Yao, X. (2019). A survey of automatic parameter tuning methods for metaheuristics. *IEEE Transactions on Evolutionary Computation*, *24*(2), 201–216.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In C. A. C. Coello (Ed.), *Learning and Intelligent Optimization* (pp. 507–523). Berlin, Heidelberg: Springer.

Hutter, F., Hoos, H. H., Stützle, T. (2007). Automatic algorithm configuration based on local search. In *Proceedings of the 22nd national conference on artificial intelligence-volume 2*, AAAI'07 (pp. 1152–1157). AAAI Press.

Jana, B., Mitra, S., & Acharyya, S. (2019). Repository and mutation based particle swarm optimization (rmpso): A new pso variant applied to reconstruction of gene regulatory network. *Applied Soft Computing*, *74*, 330–355.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*. Australia, IEEE Service Center, Piscataway, NJ, IV: Perth.

Konno, H., & Yamamoto, R. (2005). Global optimization versus integer programming in portfolio optimization under nonconvex transaction costs. *Journal of Global Optimization*, *32*, 207–219.

Konno, H., & Wijayanayake, A. (1999). Mean-absolute deviation portfolio optimization model under transaction costs. *Journal of the Operations Research Society of Japan*, *42*(4), 422–435.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2019). *Auto-WEKA: Automatic model selection and hyperparameter optimization in WEKA* (pp. 81–95). Cham: Springer.

Liang, J. J., & Suganthan, P. N. (2006). Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In *Proceeding IEEE congress on evolutionary computation* (pp. 9–16). IEEE.

Lin, S.-W., Ying, K.-C., Chen, S.-C., & Lee, Z.-J. (2008). Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert Systems with Applications*, *35*(4), 1817–1824.

Lobo, F. G., Lima, C. F., & Michalewicz, Z. (2007). *Parameter Setting in Evolutionary Algorithms* (1st ed.). Berlin: Springer.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58.

Mangasarian, O., & Han, S. (1979). Exact penalty functions in nonlinear programming. *Mathematical Programming*, *17*, 251–269.

Marinakis, Y., Marinaki, M., & Migdalas, A. (2015). Adaptive tunning of all parameters in a multi-swarm particle swarm optimization algorithm: An application to the probabilistic traveling salesman problem. In A. Migdalas & A. Karakitsiou (Eds.), *Optimization, control, and applications in the information age* (pp. 187–207). Cham: Springer.

Maron, O., & Moore, A. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Research*, *11*(1–5), 193–225.

Montero, E., Riff, M.-C., & Neveu, B. (2014). A beginner's guide to tuning methods. *Applied Soft Computing*, *17*, 39–51.

Nannen, V., & Eiben, A. E. (2007a). Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the 20th international joint conference on artifical intelligence, IJCAI'07* (pp. 975–980). Morgan Kaufmann Publishers Inc., San Francisco, CA

Nannen, V., & Eiben, A. E. (2007b). Relevance estimation and value calibration of evolutionary algorithm parameters. In M. M. Veloso (Ed.), *IJCAI 2007, proceedings of the 20th international joint conference on artificial intelligence* (pp. 1034–1039).

Nannen, V., Smit, S. K., & Eiben, A. E. (2008). Costs and benefits of tuning parameters of evolutionary algorithms. In G. Rudolph, T. Jansen, N. Beume, S. Lucas, & C. Poloni (Eds.), *Parallel problem solving from nature-PPSN X* (pp. 528–538). Berlin, Heidelberg: Springer.

Nocedal, J., & Wright, S. (2006). *Numerical Optimization—Springer series in operations research and financial engineering* (2nd ed.). Berlin: Springer.

Ozcan, E., Bas, S., & Akman, Y. (2016). The improved partıcle swarm algorıthm (pso) methods for search and rescue teams. *International Journal of Advanced Computational Engineering and Networking*, *4*(4), 22–24.

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004a). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, *8*(3), 240–255.

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004b). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, *3*(8), 240–254.

Ray, M. A. D. S. K. J., & Klepac, G. E. (2019). *Metaheuristic Approaches to Portfolio Optimization*. New York: IGI Global.

Serani, A., Leotardi, C., Iemma, U., Campana, E. F., Fasano, G., & Diez, M. (2016). Parameter selection in synchronous and asynchronous deterministic particle swarm optimization for ship hydrodynamics problems. *Applied Soft Computing*, *49*, 313–334.

Sharma, M., & Chhabra, J. K. (2019). Sustainable automatic data clustering using hybrid pso algorithm with mutation. *Sustainable Computing: Informatics and Systems*, *23*, 144–157.

Shi, Y., & Eberhart, R. (1998a). A modified particle swarm optimizer. In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No.98TH8360)* (pp. 69–73).

Shi, Y., & Eberhart, R. C. (1998b). Parameter selection in particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, & A. E. Eiben (Eds.), *Evolutionary Programming VII* (pp. 591–600). Berlin: Springer.

Shi, Y., & Obaiahnahatti, B. (1998). A modified particle swarm optimizer. In *Proceeding IEEE Congress on Evolutionary Computation* (Vol. 6, pp. 69–73).

Si, T., Jana, N., & Sil, J. (2011). Particle swarm optimization with adaptive polynomial mutation. In *Proceedings of the 2011 world congress on information and communication technologies, WICT 2011*, (pp. 12).

Si, T., Jana, N. D., & Sil, J. (2012). Pso-tuned control parameter in differential evolution algorithm. In B. K. Panigrahi, S. Das, P. N. Suganthan, & P. K. Nanda (Eds.), *Swarm, Evolutionary, and Memetic Computing* (pp. 417–424). Berlin: Springer.

Snoek, J., Larochelle, H., Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th international conference on neural information processing systems-volume 2* (pp. 2951–2959), Curran Associates Inc., Red Hook, NY.

Soler-Domínguez, A., Juan, A. A., & Kizys, R. (2017). A survey of financial applications of metaheuristics. *ACM Computing Survey*, *50*(1), 1–23.

Talbi, E., & Nakib, A. (Eds.). (2019). *Bioinspired heuristics for optimization*. Berlin: Springer.

Tang, Y., Wang, Z., & Fang, J. (2011). Feedback learning particle swarm optimization. *Applied Soft Computing*, *11*(8), 4713–4725.

Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, *85*(6), 317–325.

Trujillo, L., González, E. Á., Galván, E., Tapia, J. J., & Ponsich, A. (2020). On the analysis of hyper-parameter space for a genetic programming system with iterated f-race. *Soft Computing, 24*, 14757–14770.

Wang, H., Geng, Q., & Qiao, Z. (2014). Parameter tuning of particle swarm optimization by using Taguchi method and its application to motor design. In *2014 4th IEEE international conference on information science and technology* (pp. 722–726).

Wang, S., Li, Y., & Yang, H. (2019). Self-adaptive mutation differential evolution algorithm based on particle swarm optimization. *Applied Soft Computing*, *81*, 105496.

Winner, K., Miner, D., & desJardins, M. (2009). Controlling particle swarm optimization with learned parameters. In *Self-Adaptive and Self-Organizing Systems (SASO), 2009 3rd IEEE International Conference on* (pp. 288–290).

Wu, P., & Zhang, J. (2013). Novel particle swarm optimization for unconstrained problems. In *2013 25th Chinese control and decision conference (CCDC)* (pp. 368–372). IEEE.

Xia, X., Tang, Y., Wei, B., Zhang, Y., Gui, L., & Li, X. (2020). Dynamic multi-swarm global particle swarm optimization. *Computing*, *102*(7), 1587–1626.

Zangwill, W. (1967). Nonlinear programming with penalty functions. *Management Science*, *13*, 344–358.

Zhang, Y., Liu, X., Bao, F., Chi, J., Zhang, C., & Liu, P. (2020). Particle swarm optimization with adaptive learning strategy. *Knowledge-Based Systems*, *196*, 105789.

Zhan, Z., Zhang, J., Li, Y., & Chung, H. S. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Part B (Cybernetics)*, *39*(6), 1362–1381.

Zhan, Z.-H., & Zhang, J. (2008). Adaptive particle swarm optimization. In M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, & A. F. T. Winfield (Eds.), *Ant Colony Optimization and Swarm Intelligence* (pp. 227–234). Berlin, Heidelberg: Springer.