Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080619

Artificial Bee Colony Algorithm Hybridized with Firefly Algorithm for Cardinality Constrained Mean-Variance Portfolio Selection Problem

Milan Tuba[*] *and Nebojsa Bacanin*

Faculty of Computer Science, Megatrend University, Belgrade, Serbia

Received: 27 Dec. 2013, Revised: 28 Feb. 2014, Accepted: 01 Mar. 2014
Published online: 1 Nov. 2014

Abstract: Portfolio selection (optimization) problem is a very important and widely researched problem in the areas of finance and economy. Literature review shows that many methods and heuristics were applied to this hard optimization problem, however, there are only few implementations of swarm intelligence metaheuristics. This paper presents artificial bee colony (ABC) algorithm applied to the cardinality constrained mean-variance (CCMV) portfolio optimization model. By analyzing ABC metaheuristic, some deficiencies such as slow convergence to the optimal region, were noticed. In this paper ABC algorithm improved by hybridization with the firefly algorithm (FA) is presented. FA's search procedure was incorporated into the ABC algorithm to enhance the process of exploitation. We tested our proposed algorithm on standard test data used in the literature. Comparison with other state-of-the-art optimization metaheuristics including genetic algorithms, simulated annealing, tabu search and particle swarm optimization (PSO) shows that our approach is superior considering quality of the portfolio optimization results, especially mean Euclidean distance from the standard efficiency frontier.

Keywords: Artificial bee colony algorithm (ABC), firefly algorithm (FA), swarm intelligence, nature inspired algorthms, optimization metaheuristics, portfolio optimization, cardinality constraints.

1 Introduction

Most real-life problems can be reduced to some kind of optimization, thus optimization is one of the most applicable areas of mathematics and computer science. The difficulty of an optimization problem depends on the type of the objective function that is optimized, constraints and decision variables.

Multi-objective optimization is much more complicated than single-objective problems. The problem becomes even harder when some variables can take real, while other can take only integer values. Such mixed continuous/discrete problems usually require problem-specific search techniques in order to generate optimal, or near-optimal solution.

This research was supported by Ministry of Education, Science and Technological Development of Republic of Serbia, Grant III-44006.

1.1 Numerical optimization problems

Numerical optimization problems can be combinatorial (discrete, where variables can take only integer values) or continuous (global optimization), where continuous problems can be constrained or unconstrained (bound constrained). Portfolio optimization problem is a very important and widely researched problem in the areas of finance and economy and it belongs to the group of numerical optimization problems, with or without constants, with real and sometimes mixed variables.

Unconstrained (bound constrained) optimization is formulated as D-dimensional minimization or maximization problem:

$$min\,(or\,max)\;f(x),\,x=(x_1,x_2,x_3,...,x_D)\in S, \qquad (1)$$

[*] Corresponding author e-mail: tuba@ieee.org

where x represents a real vector with $D \geq 1$ components and $S \in R^D$ is hyper-rectangular search space with D dimensions constrained by lower and upper bounds:

$$lb_i \leq x_i \leq ub_i, \;\; i \in [1, D] \tag{2}$$

In Eq. (2) lb_i and ub_i are lower and upper bounds of the i^{th} problem component respectively.

The nonlinear constrained optimization problem in the continuous space can be formulated in the same way as in Eq. (1), but in this case $x \in F \subseteq S$ where S is D-dimensional hyper-rectangular space as defined in Eq. (2) and $F \subseteq S$ represents the feasible region defined by the set of m linear or non-linear constraints:

$$\begin{aligned} g_j(x) \leq 0, \;\; for \;\; j \in [1, q] \\ h_j(x) = 0, \;\; for \;\; j \in [q+1, m] \end{aligned} \tag{3}$$

where q is the number of inequality constraints, and $m-q$ is the number of equality constraints.

Basic versions of algorithms for constrained numerical optimization problems do not employ methods for dealing with constraints. For this reason, constraint handling techniques are usually applied in those algorithms to improve and redirect the search process towards the feasible region of the search domain. Moreover, equality constraints make optimization even harder by shrinking the feasible search space which becomes very small compared to the entire search space. To tackle such problem, equality constraints are replaced with the inequality constraints [1].

$$|h(x)| - \upsilon \leq 0, \tag{4}$$

where $\upsilon > 0$ is some small violation tolerance.

1.2 Nature-inspired metaheuristics

Deterministic algorithms are not suitable for hard, intractable optimization problems since the results cannot be obtained within an acceptable computational time. In such cases, the use of metaheuristics is more appropriate.

Metaheuristics are iterative, population based and stochastic approaches that do not guarantee the optimal solution, bet they can obtain subsatisfying suboptimal solution within reasonable computational time. Two main driving forces of metaheuristics are exploitation and exploration. Exploitation conduct search around the current best solutions, while exploration performs a random search to find the feasible region.

In the last few decades, nature became a great source of inspiration for the development of intelligent systems that can provide solutions to hard optimization problems. Following natural principles nature-inspired metaheuristics were devised.

Nature-inspired metaheuristics can roughly be divided into two categories: evolutionary algorithms (EA) and swarm intelligence. Prominent among EA are genetic algorithms (GA). GA implementations can obtain good results for many kinds of optimization problems [2].

The branch of nature-inspired algorithms which is called swarm intelligence is focused on collective behavior of some simple individuals. Social behavior of swarms of ants, bees, worms, flocks of birds and schools of fish was an inspiring source for emerging of swarm intelligence. Even though swarm system consist of relatively unsophisticated individuals, they exhibit coordinated behavior that directs swarm towards the desired goal with no central component that manages the system as a whole.

Ant colony optimization (ACO) models the social behavior of ants in finding the shortest paths between their nest and the food source. The corner stone of the ACO is ant's ability to deploy a substance called pheromone in order to mark discovered path. ACO is one of the oldest members of swarm intelligence family [3]. This metaheuristic was successfully applied to combinatorial [4], as well as on continuous optimization problems [5], [6], [7]. Particle swarm optimization (PSO) is another older swarm intelligence algorithm that simulates social behavior of fish schooling or bird flocking. PSO was successfully applied to many single-objective and multi-objective optimization problems. Glowworm swarm optimization algorithm was recently applied to constrained engineering design problems [8].

Metaheuristic that mimics the human search process based on human memory, reasoning, past experience and interactions is seeker optimization algorithm (SOA). This relatively novel method showed good performance in solving global numerical optimization problems [9] and is continuously being improved [10].

Cuckoo search (CS) is another new iterative approach that models search process by employing Levy flights (series of straight short and long flight paths with sudden 90 degrees turn). It was first proposed by Yang and Deb [11] and proven to be a robust optimization technique [12], obtaining satisfying results in real-life optimizations like image thresholding [13] based on entropy objective function [14].

1.3 Artificial bee colony (ABC) improvement

In this paper we propose Artificial Bee Colony (ABC) algorithm hybridized with Firefly Algorithm (FA) for cardinality constrained mean-variance (CCMV) portfolio optimization problem. ABC was originally proposed by Karaboga for continuous optimization problems [15], while FA is among the latest swarm intelligence algorithms proposed by Yang [16].

Our implementation of the hybridized ABC solution is aimed to overcoming weaknesses of the original ABC for constrained optimization problems such as CCMV portfolio problem. By studying the ABC algorithm, we noticed a deficiency during the solution search process. Exploitation is not intensive enough and the algorithm converges slowly to the optimal region of the search space. After significant number of cycles, when the optimal solution is almost found, this deficiency is even more emphasized.

Additionally, the exploitation-exploration balance is not well adjusted for this application in the original ABC approach. In early cycles, scouts perform exploration which is necessary for finding feasible search space region, however is not well balanced with exploitation. In the late cycles, with the assumption that the search has converged to the optimal region, more exploitation power is needed.

By analyzing search process of the firefly algorithm (FA), we noticed that this metaheuristic employs more intensive exploitation. In the FA, more variables are being utilized when performing search than in the ABC. Also, ABC uses modification rate (MR) parameter, and the solutions in the population are not being modified in every cycle.

In order to improve both, the exploitation process, and exploration-exploitation balance, we encapsulated FA search equation in the employed bee phase. Our hybridized metaheuristic performs ABC or FA search depending on the firefly search trigger (FST) parameter. By integrating FA search, exploitation is intensified and better exploitation-exploration balance is established.

In this way, by integrating FA search into the ABC, we derived enhanced hybridized metaheuristic for cardinality constrained mean-variance (CCMV) portfolio optimization problem which is named artificial bee colony with firefly search method (ABC-FS).

The rest of the paper is organized as follows. Literature review is given in Section 2, where we enlist implementations of metaheuristics for CCMV portfolio problems found in the literature survey. Section 3 presents mathematical formulations of portfolio optimization problems. Original ABC approach for constrained optimization is described in Section 4. In Section 5, we give detail description and analysis of our ABC-FS approach. Parameter settings and experimental results are shown in Section 6, while conclusion and final remarks are given in Section 7.

2 Literature Review

In this subsection brief overview is given of some metaheuristic implementations for portfolio optimization problem found in the literature. As a result of literature survey, it can be concluded that portfolio selection problem was not much researched using nature inspired metaheuristics, and to the extend it was researched, mostly genetic algorithm (GA) implementations were used. Also, it was observed that there are only few swarm intelligence algorithms adopted for portfolio optimization. In this section, some of the most important metaheuristic implementations for portfolio optimization problem are discussed.

Many papers show solving portfolio optimization problem using non-dominating Sorting Genetic Algorithm (NSGA). First version of NSGA algorithm was proposed by Deb et al. [17]. Difference between GA and NSGA is the redefinition of the selection operator. Second version, NSGA-II was also proposed by Deb et. al [18]. New version improves the convergence and the spread of the solutions in population.

Lin et al. [19] considered a MV portfolio model with minimum transaction lots (MTL), fixed transaction costs (TC) and linear constraints on capital invested similar to the holding weights constraints. NSGA-II based algorithm with integer encoding was proposed to tackle this problem. The results were satisfying.

Streichert et al. [20] implemented NSGA with real value and integer encoding for solving MV portfolio model constrained to cardinality, buy-in thresholds and MTL constraints. By examining results of preliminary experiments, the authors noticed that the efficient frontier of the portfolio optimization problem is generally composed of a restrictive number of the initial available assets, and outlined the analogy with the one-dimensional binary knapsack problem. Taso and Lui [21] applied modified NSGA-II to the Mean-Var portfolio problem. They considered only budget constraint in problem formulation. They changed random initialization of solutions in original NSGA-II implementation and endorsed the method that spots the non-dominated solution set given a population of chromosomes. This approach performs well in portfolio optimization.

Hybrid GA approach was devised for portfolio optimization problem formulation whose purpose is to duplicate within a target portfolio the behavior of a stock market index chosen as a benchmark. Problem's objective function is chosen for minimizing the tracking error variance. The algorithm was tested on DAX indexes benchmark using one year of daily closing prices [22].

In [23] optimization of MV portfolio problem with cardinality and holding weights constraints is presented. Comparative analysis of efficient frontiers of three algorithms is given. Performance of GA, tabu search (TS) and simulated annealing (SA) is compared on a small example problem which comprises four assets of FTSE index with a cardinality fixed to two. Computer time and different percentage errors are used as a comparison indicators. Testing results showed that for unconstrained portfolio optimization GA gives the best approximation with an almost zero mean percentage error, while for cardinality constrained problems, none of the heuristics is uniformly superior [23].

Soleimani et al. [24] showed GA with RAR crossover operator for solving MV portfolio problem where cardinality constraints, MTL and constraints on sector capitalization are taken account. Besides RAR operator, the authors also employed in their approach a selection operator wherein half of the population is conducted to the following generation by choosing the fitter chromosomes, and the other half is composed of offspring chromosomes. GA was compared to the LINGO results on a small assets problem. Results showed that the error differences of both approaches are minimal with no more than 3 percent, but GA performed in a much less time than LINGO. The second experiment on a data set of 2,000 assets showed the efficiency of GA, in both parameters, computational time and risk error.

Among other metaheuristics for portfolio problem, one approach based on neural networks (NN) should be distinguished [25]. In this paper, particular NN, the Hopfield network is used to trace out the efficient frontier for cardinality constrained portfolio problem. This approach was compared to several state-of-the-art metaheuristics for the same problem and showed good performance.

As mentioned above, there are only few swarm intelligence approaches for portfolio optimization. Deng and Li presented ant colony optimization (ACO) for solving he cardinality constraints Markowitz MV portfolio model [26]. Numerical solutions are obtained for five analyses of weekly price data for the Hang Seng, DAX, FTSE 100, S&P and Nikkei indexes. The test results indicate that the ACO is much more effective than PSO, especially for low-risk investment portfolios.

Haqiqi and Kazemi [27] proposed ant colony optimization (ACO) for solving MV portfolio model. The performance of ACO metaheuristic was compared with the $frontcon$ function of MATLAB software as an exact method. The test data set were monthly prices for three years from Teheran stock exchange. The results show that proposed ACO approach is reliable, but not preferred to an exact method.

Cura showed PSO approach to cardinality constrained MV portfolio optimization [28]. The test data set is the weekly prices from March 1992 to September 1997 from the following indexes: Hang Seng in Hong Kong, DAX 100 in Germany, FTSE 100 in UK, S&P 100 in USA and Nikkei in Japan. The results of this study are compared with those of genetic algorithms, simulated annealing and tabu search approaches, and showed that PSO has potential in portfolio optimization.

Zhu et al. [29] presented PSO algorithm for non-linear constrained portfolio optimization with multi-objective functions. The model is tested on various restricted and unrestricted risky investment portfolios and a comparative study with GA is showed. PSO demonstrated high computational efficiency in constructing optimal risky portfolios and can be compared with other state-of-the-art algorithms.

ABC algorithm for mixed quadratic and integer programming problem of cardinality constrained MV portfolio model was presented by Wang et al [30]. Some modifications of classical ABC algorithm for constrained optimization problems were adopted. The approach was tested on a standard benchmark data set and proved to be a robust portfolio optimizer. Also, a hybridized ABC with differential evolution (DE) metaheuristic was proposed by the same authors for solving cardinality constrained MV problem [31]. In the hybrid ABC algorithm, a new search scheme and obsolete rules are presented to improve convergent speed of the algorithm.

One of the first implementations for portfolio optimization problem by the firefly algorithms was developed by Tuba et al. [32]. Framework for solving this problem was devised. Metaheuristics was tested on a five assets data set. FA proved to be robust and effective technique for portfolio problem.

3 Models for Portfolio Optimization

The basic guideline in making financial investments decisions is diversification, where investors invest into different types of assets. By investing in portfolios, rather than in single assets (or securities), individuals and institutions are able to dampen the risk by diversification of the investments, with no negative effect on expected returns. Thus, the portfolio diversification minimizes investors' exposure to the risks [33], while maximizing returns on portfolios [34].

In its basic form, portfolio optimization is concerned with selecting the portfolio of securities that minimizes the risk subject to the constraint of guaranteeing a given level of returns [35]. This problem belongs to the group of multi-objective optimization problems. Many methods were devised for solving this kind of problem. One essential method which can be divided into two sub-types tackles the problem by transforming multi-objective portfolio model into a single-objective.

The first sub-type selects one important objective function for optimization, while the remaining objective functions are treated as constraints. This method is defined by Markowitz and it is called the standard mean-variance (MV) model [36]. It was first formulated in seminal paper in 1952, where the author rejects the hypothesis that investors wish to maximize expected returns because this criterion does not imply that a diversified portfolio is preferable to a non-diversified one [35]. MV model's basic assumptions are that the investors are rational with either multivariate normally distributed asset returns, or, in the case of arbitrary returns, a quadratic utility function [37]. If those assumptions hold, then the optimal portfolio for the investor lies on the mean-variance efficient frontier.

In Markowitz's MV model, the selection of risky portfolio is modeled as objective function, while the mean return of an asset is considered to be one of the constraints [38]. Mathematical formulation is as follows:

$$min\ \sigma_{R_p}^2 = \sigma_p^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} \omega_i \omega_j Cov(\overline{R}_i \overline{R}_j) \tag{5}$$

Subject to

$$\bar{R_p} = E(R_p) = \sum_{i=1}^{N} \omega_i \bar{R_i} \geq R \tag{6}$$

$$\sum_{i=1}^{N} \omega_i = 1 \tag{7}$$

$$\omega_i \geq 0,\ \forall i \in (1, 2, \ldots N) \tag{8}$$

where N is the number of available assets, $\bar{R_i}$ is the mean return on an asset i and $Cov(\bar{R_i}\bar{R_j})$ is covariance of returns of assets i and j respectively. Weight variable ω_i is used as a control parameter that defines the proportion of the capital that is invested in asset i, and constraint in Eq. (7) ensures that the whole available capital is invested. In this formulation, the objective is to minimize the portfolio risk σ_p^2, for a given value of portfolio expected return $\bar{R_p}$.

In the shown MV model, weight variables (ω) are real and they are in range between 0 and 1, as they represent the fraction of available money to invest in an asset. This choice is quite straightforward and has the advantage of being independent of the actual budget. It should be noted that the Markowitz model can be considered as the most simple formulation of portfolio optimization problem.

The second sub-type refers to the construction of only one evaluation function that models portfolio optimization problem. It is often called in the literature single-objective function model. This method encompasses two distinct models: efficient frontier and Sharpe ratio model [29].

The main goal in the efficient frontier model is to find the different objective function values by varying desired mean return R. For this purpose, new parameter $\lambda \in [0, 1]$, which is called risk aversion indicator, is introduced [29]. In this case, the model is approximated to only one objective function:

$$min\ \lambda [\sum_{i=1}^{N}\sum_{j=1}^{N} \omega_i \omega_j Cov(\bar{R_i}\bar{R_j}] - (1-\lambda)[\sum_{i=1}^{N} \omega_i \bar{R_i}] \tag{9}$$

subject to

$$\sum_{i=1}^{N} \omega_i = 1 \tag{10}$$

$$\omega_i \geq 0,\ \forall i \in (1, 2, \ldots N) \tag{11}$$

Parameter λ controls the relative importance of the mean return to the risk for the investor. When λ is 0, mean return of the portfolio is maximized regardless of the risk. Oppositely, when λ has value of 1, risk of the portfolio is being minimized regardless of the mean return. Thus, when the value of λ rises, the relative importance of the risk to the investor increases, and significance of the mean return decreases, and vice-versa.

When the value of λ changes, objective function's value alters. The reason for this change is that the objective function is composed of the mean return value and the variance (risk). The dependencies between changes of λ and the mean return and variance intersections are shown on a continuous curve which is called efficient frontier in the Markowitz theory [36]. Since each point on this curve indicates an optimum, portfolio optimization problem is considered as multi-objective, but λ transforms it into single-objective optimization task.

Sharpe ratio (SR) model uses the information from mean and variance of an asset [39]. This simple model is risk-adjusted measure of mean return and can be described with the following expression [39]:

$$SR = \frac{R_p - R_f}{StdDev(p)}, \tag{12}$$

where p denotes portfolio, R_p is the mean return of the portfolio p, and R_f is a test available rate of return on a risk-free asset. $StdDev(p)$ is a measure of the risk in portfolio (standard deviation of R_p). By adjusting the portfolio weights w_i, portfolio's Sharpe ratio can be maximized.

In the models presented so far, we showed only basic problem formulations that do not consider real-world factors and limitations and cannot be applied in practice. These real-world factors and limitations include the existence of transaction costs, sectors with high capitalization and taxation, specifications of legal and economic environment, finite divisibility of the assets to select, etc. [40]. Thus, additional constraints can be applied to make portfolio optimization problem more realistic. For example, budget, cardinality, transaction lots and sector capitalization constraints were successfully applied in solving portfolio optimization problem using PSO metaheuristic in [38]. The minimum transaction lots constraint assures that each asset can only be purchased in a certain number of units. With the applied transaction lots constraint, classical portfolio optimization problem becomes a combinatorial optimization problem whose feasible region is not continuous. Sector capitalization constraint refers to the fact that the investors tend to invest in the assets that belong to the sectors where higher value of market capitalization can be obtained. Investing in such way, risk is reduced. The importance of this constraint is discussed in [24].

If all the above mentioned additional portfolio optimization constraints are taken account, new portfolio optimization problem can be established [38]. This model is named extended MV model and it is classified as a quadratic mixed-integer programming model which can be solved only by employing heuristics techniques. Extended MV model can be formulated as follows:

$$min\ \sigma_{R_p}^2 = \sigma_p^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} \omega_i \omega_j Cov(\bar{R_i}\bar{R_j}) \tag{13}$$

where

$$\omega_i = \frac{x_i c_i z_i}{\sum_{j=1}^{N} x_j c_j z_j}, \; i = 1, \ldots, N \tag{14}$$

$$\sum_{i=1}^{N} z_i = M \leq N, \; M, N \in N, \; \forall i = 1, \ldots N, \; z_i \in \{0,1\} \tag{15}$$

subject to

$$\sum_{i=1}^{N} x_i c_i z_i \bar{R}_i \geq BR \tag{16}$$

$$\sum_{i=1}^{N} x_i c_i z_i \leq B \tag{17}$$

$$0 \leq B_{low_i} \leq x_i c_i \leq B_{up_i} \leq B, \; i = 1, \ldots N \tag{18}$$

$$\sum_{i_s} W_{i_s} \geq \sum_{i_{s'}} W_{i_{s'}} \tag{19}$$

$$\forall y_s y_{s'} \neq 0, \; s, s' \in \{1, \ldots S\}, \; s < s'$$

where

$$y_s = \begin{cases} 1 & \text{if } \sum_{i_s} z_i > 0 \\ 0 & \text{if } \sum_{i_s} z_i = 0 \end{cases} \tag{20}$$

where M represents the number of selected assets among possible N assets. B is the total available budget, while B_{low_i} and B_{up_i} are lower and upper limits of the budget that can be invested in asset i respectively. S denotes the total number of sectors in the market. c_i is the minimum transaction lot for asset i, and x_i denotes the number of c_i' that is purchased. According to this, $x_i c_i$ are integer values that show the units of asset i in the portfolio.

Decision variable z_i is defined for modeling cardinality constraint. z_i is equal to 1 if an asset i is present in the portfolio. Otherwise, it is equal to 0. Eq. (15) represents the cardinality constraint and inequality in Eq. (16) is the same as in Eq. (6). In order to make the search process easier, budget constraint, Eq. (17) is converted to inequality. Eq. (18) shows lower and upper bounds of budget constraint.

Sector capitalization constraint improves decisions of portfolio's structure by emphasizing investments in assets that belong to the sector with higher capitalization value. The assets that belong to the sector with more capitalization should have more share in the final portfolio. This constraint is held only if securities from the corresponding sectors are selected [38]. Eq. (19) models sector capitalization constraint. Despite of the fact that a certain sector has high capitalization, security from this sector that has low return and/or high risk must be excluded from the final portfolio's structure. To make such exclusion, variable y_s is defined and it has a value of 1 if the corresponding sector has at least one selected asset, and 0 otherwise. In Eq. (19), i_s is a set of assets which can be found in sector s. Sectors are sorted in descending order by their capitalization value. Sector 1 has the highest capitalization value, while sector S has the lowest value.

In the literature other constraints can be found. One of them is 5-10-40 constraint which is based on the §60(1) of the German investment law [41]. This constraint defines upper limit of each individual asset and for the sum of all ”heavyweight” in the portfolio. It actually means that the securities of the same issuer are allowed to the amount of 5% of the net asset value of the mutual fund [41]. They are allowed to amount to 10%, however, if the total share of all assets with a share between 5% and 10% is less than 40% of the net asset value [41].

In this paper, for testing purposes, we use model which employs some of the constraints that can be found in the extended MV formulation. This study uses cardinality constrained mean-variance model (CCMV) which is derived from the standard Markowitz's and the efficiency frontier models. CCMV formulation is:

$$min \; \lambda [\sum_{i=1}^{N} \sum_{j=1}^{N} x_i x_j \sigma_{i,j}] - (1-\lambda)[\sum_{i=1}^{N} x_i \mu_i] \tag{21}$$

Subject to

$$\sum_{i=1}^{N} x_i = 1 \tag{22}$$

$$\sum_{i=1}^{N} z_i = K \tag{23}$$

$$\varepsilon_i z_i \leq x_i \leq \delta_i z_i, \; z \in \{0,1\}, \; i = 1, 2, 3, \ldots N \tag{24}$$

As mentioned above, λ is risk aversion parameter, x_i and x_j are weight variables of assets i and j respectively, $\delta_{i,j}$ is their covariance, and μ_i is i-th asset's return. K is the desired number of assets that will be included in the portfolio. Decision variable z_i controls whether the asset i will be included in portfolio. If its value is 1, asset i is included, and if the value is 0, asset i is excluded from the portfolio. ε and δ are lower and upper bounds of the asset that is included in portfolio and they make sure that the asset's proportion in the portfolio is within the predefined range.

From CCMV formulation it can be seen that this problem belongs to the group of mixed quadratic and integer programming problem. It employs both, real and integer variables with equity and inequity constraints.

4 Original ABC Algorithm Implementation for Constrained Optimization

The artificial bee colony (ABC) algorithm was designed for numerical optimization problems and it was inspired by the foraging behavior of honey bees [15]. Since the performance of metaheuristic algorithms depend on the number and the choice of parameters, the main advantages of the ABC algorithm are derived from the fact that the algorithm uses only 3 control parameters: colony size, maximum cycle number and limit.

In this paradigm, three types of artificial bees performs search. Each type of bee has its particular role in a search process. This algorithm proves to be robust and capable of solving high dimensionality problems [42], [43], [44].

ABC algorithm utilizes three classes of artificial bees: employed bees, onlookers and scouts. Employed bees make half of a colony. In the ABC metaheuristic, food source represents possible problem solution. There is only one employed bee per each food source. Employed bee performs search process by examining solution's neighborhood. Onlooker chooses food source for exploitation based on the information which they gain from employed bees. If a food source does not improve for a predetermined number of cycles, scouts replace that food source with a new one which is chosen randomly. Limit parameter controls this process. Thus, in the ABC algorithm onlooker and employed bees are responsible for the exploitation process, while scouts take care of the exploration.

The main difference between ABC and other swarm intelligence algorithms is based on the fact that the possible solutions are represented by the food sources, not the individuals in the population. ABC algorithm, as an iterative algorithm, starts by associating each employed bee with randomly generated food source (solution). Each solution x_i $(i = 1,2,...SN)$ is a D-dimensional vector, where SN denotes the size of the population, and D represent number. Initial population of candidate solutions is created using following expression:

$$x_{i,j} = lb_j + rand(0,1) * (ub_j - lb_j), \tag{25}$$

where $x_{i,j}$ is the j-the parameter of th i^{th} solution in the population, $rand(0,1)$ is a random real number between 0 and 1, and ub_j and lb_j are upper and lower bounds of the j^{th} parameter respectively.

There are many formulations of fitness function, but in most implementations, for maximization problems, fitness is simply proportional to the value of objective function, while for the minimization problems, the following expression is used:

$$fitness_i = \begin{cases} \frac{1}{objFun_i}, \; if\, objFun_i > 0 \\ 1 + |objFun_i|, \; otherwise \end{cases} \tag{26}$$

Each employed bee discovers a food source in its neighborhood and evaluates its fitness. Discovery of a new, neighborhood solution is simulated with the following expression:

$$v_{i,j} = \begin{cases} x_{i,j} + \phi * (x_{i,j} - x_{k,j}), R_j < MR \\ x_{i,j}, otherwise \end{cases} \tag{27}$$

where $x_{i,j}$ is j^{th} parameter of the old solution i, $x_{k,j}$ is j^{th} parameter of a neighbor solution k, ϕ is a random number between 0 and 1, and MR is modification rate. MR is a control parameter of ABC algorithm.

If the fitness of the new solution is higher than the fitness of the old one, employed bee continues exploitation process with the new food source, otherwise it retains the old one. Employed bees share information about fitness of food source with onlookers, and onlookers select a food source i with a probability that is proportional to the solution's fitness taking into account constraint violations (CV):

$$p_i = \begin{cases} 0.5 + (\frac{fitness_i}{\sum_{i=1}^{SN} fitness_i}), \; if\, solution\, is\, feasible \\ (1 - \frac{CV}{\sum_{i=1}^{SN} CV}) * 0.5, \; if\, solution\, is\, infeasible \end{cases} \tag{28}$$

where CV is calculated using:

$$CV_i = \sum_{g_j(x_i)>0} g_j(x_i) + \sum_{j=q+1}^{m} h_j(x_i) \tag{29}$$

Taking into account all the above mentioned, pseudo-code for the ABC algorithm for constrained problems is given below.

```
Initialize the population of solutions using Eq. (25)
Evaluate the population using Eq. (26)
$cycle = 1$
**repeat**
  Produce new solutions for the employed bees by
    using Eq. (27), and evaluate them
  Apply selection process between old and new
    solutions based on the Deb's method [45], [46]
  Calculate the probability values $p_i$ for the
    solution $x_i$ using Eq. (29)
  For each onlooker bee, produce a new solution $v_{ij}$
    using (27) in the neighborhood of the solution which
    is selected according to the probability value $p_i$
  Apply selection process between new solution
    $v_i$ and old solution $x_i$ by employing Deb's method
  Determine the abandoned solutions by using limit
    parameter for the scout. If they exist, replace them
    with new randomly produced solutions by Eq. (25)
  Memorize the best solution achieved so far
  $cycle = cycle + 1$
**until** $cycle = MCN$
```

5 Proposed Hybridized ABC-FS Metaheuristic for CCMV Portfolio Problem

In order to enhance the performance of the original ABC metaheuristic, we adopted search method procedure from the FA algorithm. This search strategy improves exploitation and convergence speed of the basic ABC implementation. Also, for solving CCMV portfolio problem, some modifications of original ABC algorithm were necessary, especially for constraints handling.

5.1 Initialization phase

At the initialization step, the algorithm generates random population of *SN* positions (food sources) x_i. For random initialization, ABC-FS employs the same expression as in the original ABC Eq. (25). Moreover, decision variable $z_{i,j}$ $(i = 1, \ldots SN, j = 1, \ldots N)$ is also initialized for each food source *i*. *N* is the number of potential assets in portfolio. Thus, the number of dimensions of each potential solution is $2N$. z_i is binary array, and when an asset is included in portfolio, it's value is 1, otherwise it is 0.

At the initialization stage, decision variables are generated randomly using:

$$z_{i,j} = \begin{cases} 1, if\, \phi < 0.5 \\ 0, if\, \phi \geq 0.5 \end{cases} \tag{30}$$

where ϕ is random real number between 0 and 1.

At this stage, *limit* counter that controls whether the solutions should be abandoned is set to 0 for all candidate solutions in the populations.

Inspired with the similar approach proposed in [28], we used the arrangement algorithm that guarantees the feasibility of solutions. This algorithm is first applied at the initialization stage of our ABC-FS. In this algorithm, *i* is the current solution that consists of: *Q* the distinct set of K_i^* assets in the i^{th} solution, $z_{i,j}$ is the decision variable of asset *j* and $x_{i,j}$ is the weight proportion for asset *j*. Arrangement algorithm pseudo-code is shown below.

```
while($K_i^* < K$)
  select random asset $j$ such that $j \notin Q$
  $z_{i,j} = 1, Q = Q \cup [j], K_i^* = K_i^* + 1$
end while
while($K_i^* > K$)
  select random asset $j$ such that $j \in Q$
  $z_{i,j} = 1, Q = Q - [j], K_i^* = K_i^* - 1$
end while
while(true)
  $\theta = \sum_{j \in Q} x_{i,j}, x_{i,j} = x_{i,j}/\psi, \eta =$
  $= \sum_{j \in Q} max(0, x_{i,j} - \delta_i), \phi = \sum_{j \in Q} max(0, \eta_j - x_{i,j})$
  if($\eta = 0\, and\, \phi = 0$) then exit algorithm
  for $j = 1$ to $N$
    if ($z_{i,j} = 1$) then
      if ($x_{i,j} > \delta_j$) then $x_{i,j} = \delta_j$
      if ($x_{i,j} < \varepsilon_j$) then $x_{i,j} = \varepsilon_j$
    end if
  end for
end while
```

As can be seen from the presented pseudo-code, for the constraint $\sum_{i=1}^{N} x_i = 1$ we set $\psi = \sum_{j \in Q} x_{i,j}$ and put $x_{i,j} = x_{i,j}/\psi$ for all assets that satisfy $j \in Q$. The same approach for satisfying this constraint was used in [28]. To ensure that each asset's proportion is within predefined lower and upper bounds, ε and δ respectively, we used: $if\; x_{i,j} > \delta_{i,j}\; then\; x_{i,j} = \delta_{i,j}$ and $if\; x_{i,j} < \varepsilon_{i,j}\; then\; x_{i,j} = \varepsilon_{i,j}$. We emphasis that we did not use *c*-value based approach for putting out and in assets in the portfolio as in [28]. In our algorithm, assets are being added and removed from the portfolio randomly.

5.2 Fitness calculation and employed bees phase

Fitness is calculated as in the original ABC implementation according to Eq. (26). As in the original ABC metaheuristic, the number of employed bees is equal to the number of food sources. In each algorithm's iteration, an employed bee finds new food source and evaluates its fitness. Process of finding new food source is defined as:

$$z_{i,j}^{new} = round(\frac{1}{1 + e^{-z_{i,j} + \phi_{i,j}(z_{i,j} - z_{k,j}}}) - 0.06) \tag{31}$$

$$x_{i,j}^{new} = \begin{cases} x_{i,j} + \phi_{i,j} * (x_{i,j} - x_{k,j}),\, if\, R_j < MR \wedge z_{i,j}^{new} = 1 \\ x_{i,j}, otherwise \end{cases} \tag{32}$$

where $z_{i,j}$ is a decision variable of the j-th parameter of the old solution, $z_{k,j}$ is decision variable of j^{th} parameter of the neighbor solution. $x_{i,j}$ is j^{th} parameter of the old solution *i*, $x_{k,j}$ is j^{th} parameter of a neighbor solution *k*, $\phi_{i,j}$ is a random number between 0 and 1, and *MR* is modification rate. *MR* is a control parameter of the ABC algorithm.

It should be noticed that the decision variables in the employed bee phase are generated differently than in the initialization phase Eq. (30)

As mentioned in the *Introduction*, we incorporated search procedure form the FA metaheuristic. FA was first proposed for unconstrained optimization [16]. FA has been adapted for solving various numerical optimization and practical optimization problems.

Our inspiration for enhancement of the search process came from the firefly's flashing behavior that is used to improve search for the optimal solution. Thus, in ABC-FS implementation, the employed bee phase is modified by hybridization with the firefly search. We introduced another search expression which is applied for each solution's parameter [16]:

$$x_i^{new} = x_i + \beta_0 * e^{-\gamma * r_{i,k}^2}(x_k - x_i) + \alpha * (rand - 0.5), \quad (33)$$

where β_0 is attractiveness at r=0, γ presents the variation of attractiveness, α is randomization parameter, *rand* is random number uniformly distributed between 0 and 1, and $r_{i,k}$ is distance between food sources i and k.

The distance between food sources i and k is calculated using Cartesian distance:

$$r_{i,k} = ||x_i - x_k|| = \sqrt{\sum_{j=1}^{D}(x_{i,j} - x_{k,j})^2}, \quad (34)$$

where D is the number of problem variables.

For most cases, it can be taken $\beta_0 = 0$ and $\alpha \in [0,1]$. In the ABC-FS, α is dynamic parameter which is adjusted during the algorithm's run. The details will be presented in Section 4. γ is extremely important in determination of the speed of the convergence of the search process.

It should be noted that in our algorithm, the attractiveness is modeled with the fitness function. The higher the fitness, the higher the attractiveness and vice-versa.

To control the search process, we introduced additional parameter, firefly search trigger (FST) which is adjusted within the range $[0,1]$. If $\kappa \leq FST$, then the ABC search is performed, otherwise, FA search mechanism is triggered. κ is random number uniformly distributed between 0 and 1. We note that when the FA search is performed, the decision variables are calculated the same way Eq. (31). In our implementation, we empirically determined FST = 0.5, so with equal probably ABC or FA search will be performed.

After the employed bee founds new solution, a selection between old and new solution is performed based on the Deb's rules [46]. For this selection, constraint violations CV are being calculated using Eq. (29). If the old solution cannot be improved, its counter (number of trials) is incremented by one. If the new solution is selected, count is reset to 0.

5.3 Onlooker bee phase

When all employed bees have finished their search, they share information about food source's fitness with the onlooker bees. The onlookers chose food source with probability p_i calculated using Eq. (28). ABC-FS uses fitness proportional roulette wheel selection. Food sources with higher p_i have greater chances for being selected. After that the onlooker tries to improve selected solution by using Eq. (31) and Eq. (32).

If onlooker could not improve old food source (x_i, z_i), its counter is incremented by one, otherwise, new solution (x_i^{new}, z_i^{new}) is chosen and the counter is set to 0. In the onlooker phase, we do not use FA search equation.

5.4 Scout phase

After all employed bees and onlooker bees complete their search, all food source that have reached the *limit* are abandoned and bees that exploited them become scouts. Scouts perform exploration by replacing the old food source with a random one. Random solution is generated using Eq. (25) and Eq. (30).

Pseudo-code of the ABC-FS metaheuristic for CCMV problem is shown below.

```
Initialize the population of solutions
  $z_{i,j}, x_{i,j}, i = 1,2,3,...SN, j = 1,2,3,...N$ by using
  Eq. (25) and Eq. (30)
Apply arrangement algorithm
Evaluate the population using (26)
$cycle = 1$
**repeat**
  generate decision variable for new solution using
    Eq. (31)
  **if** ($\kappa \leq FST$) produce new solutions for the employed
    bees with ABC search using Eq. (32), otherwise
    generate new solution by employing FA search with
    Eq. (33)
  Evaluate new solution
  Apply selection process between new solution
    $(x_i^{new}, z_i^{new})$ and old solution $(x_i, z_i)$ by employing
    Deb's method [45], [46]
  Calculate the probability values $p_i$ for the solution $x_i$
    using Eq. (29) and Eq. (28)
  For each onlooker bee, produce a new solution using
    Eq. (31) and Eq. (32) in the neighborhood of the
    solution which is selected according to the
    probability value $p_i$
  Apply selection process between new solution
    $(x_i^{new}, z_i^{new})$ and old solution $(x_i, z_i)$ by employing
    Deb's method [45], [46]
  Determine the abandoned solutions by using *limit*
    parameter for the scout. If they exist, replace them
    with new randomly produced solutions by Eq. (25)
  Memorize the best solution achieved so far
  $cycle = cycle + 1$
**until** $cycle = MCN$
```

In the shown pseudo-code, SN is the number of food sources (solutions) in the population, N is the number of assets in the set, and MCN is maximum number of algorithm's iterations.

6 Algorithm Settings and Experimental Results

In this section, we first present and analyze ABC-FS parameter setup. Later, we show experimental results on a standard benchmark data set and a comparative analysis with other state-of-the-art metaheuristics.

6.1 Parameter settings

For the sake of better comparative analysis, we set all algorithm parameters like in [28]. *SN*, the number of food sources (solutions) in the population was calculated using:

$$SN = 20\sqrt{N}, \tag{35}$$

where N is the number of assets in portfolio. *MCN* maximum number of cycles (iterations) was set to:

$$MCN = \frac{1000N}{SN} \tag{36}$$

Exploration and exploitation are two fundamental elements of all EAs that drive and direct the evolution process towards optimum and/or convergence. Exploration refers to visiting new regions of the search space, while exploitation explores the search space within the neighborhood of previously visited points.

Modification rate (*MR*) and *limit* parameters control exploitation-exploration trade-off in the ABC approach. *MR* parameter is set to 0.8, and *limit* is set according to the following expression [43]:

$$limit = \frac{MCN}{SN} = \frac{\frac{1000N}{SN}}{20\sqrt{N}} \tag{37}$$

Since *MR* and *limit* control the exploitation-exploration balance, empirical tests showed that this is proper parameter adjustment.

We empirically established an equal selection probability between ABC and FA search procedures, so a parameter *FST* is set to 0.5.

FA search parameter α process is set to 0.5, but it is being gradually decreased from its initial value according to the Eq. (38).

$$\alpha(t) = (1-(1-((10^{-4}/9)^{1/cycle})))*\alpha(t-1) \tag{38}$$

In the ABC-FS implementation, besides the adoption of arrangement algorithm, we used Eq. (4) and violation limit υ for handling constraints. It is very important to chose the right value for υ. If the chosen value is too small, the algorithm may not find feasible solutions, and otherwise the results may be far from the feasible region [43].

The promising approaches for handling equality constraints include dynamic, self-adaptive tolerance adjustment [47]. Exploration is improved by exploring a larger search space than the initial one. One of the best practices is to start with a relatively large υ value, which is gradually decreased through the iterations of the algorithm. We used the following dynamic settings for the υ:

$$\upsilon(cycle+1) = \frac{\upsilon(cycle)}{dec} \tag{39}$$

where t is the current cycle, and *dec* is a value slightly larger than 1 ($dec \sim 1$). For handling equality constraints, we set initial value for υ to 1.0, *dec* to 1.001 and the threshold for υ to 0.0001 like in [43].

For generating heuristics efficient frontier we used different λ values from 0 to 1 with $\Delta\lambda$ = 0.02 as in [28], which gives the number of algorithm's runs ξ = 51.

We also set the number of assets that will be included in portfolio K to 10, lower asset's weight ε to 0.01 and upper asset's weight δ to 1.

Below, we present again short ABC-FS pseudo-code, but this time emphasizing parameter adjustments.

```
λ = 0
while(λ ≤ 1)
  SN = 20√N
  Set portfolio problem parameters K, υ and δ
  InitializationPhase()
  ArrangementAlgorithm()
  EvaluatePopulation()
  Set values for MR and limit
  Set initial values for υ and α
  MCN = 1000N/SN
  for(cycle=1 to MCN)
    for(i=1 to SN)
      if (κ ≤ FST) EmployedBeePhase()
        by ABC search
      else
      EmployedBeePhase() by FA search
      Apply Selection between old and new solution
        using Deb rules
    end for
    for(i=1 to SN)
      OnlookerBeePhase()
      Apply Selection between old and new solution
        using Deb rules
    end for
    ScoutBeePhase()
    ArrangementAlgorithm()
    MemorizeBestSolution()
    Recalculate values for υ and α
    cycle++
  end for
  λ=λ + Δλ
end while
```

In order to make better distinction between parameter types, we divided parameters into following groups:

Table 1 ABC-FS parameters

Parameter	**Value**
ABC-FS global parameters	
Number of food sources (*SN*)	depends on *N*
Number of cycles (*MCN*)	depends on *SN*
Limit (*limit*)	depends on *SN* and *MCN*
Modification rate (*MR*)	0.8
Firefly search trigger (*FST*)	0.5
FA search parameters	
Initial value for randomization parameter α	0.5
Attractivnes at r=0 β_0	0.2
Absorption coefficient γ	1.0
Portfolio problem parameters	
Number of assets (N)	depends on the problem
No. of assets in portfolio (K)	10
Initial value of risk aversion (λ)	0
Different λ values (ξ)	51
Lower asset's weight (ε)	0.01
Upper asset's weight (δ)	1.0
Constraint-handling parameters	
Initial violation tolerance (υ)	1.0
Decrement (*dec*)	1.002

Table 2 ABC-FS benchmark specific parameters

Parameter	**Value**
Hang Seng index with 31 assets	
Number of food sources (*SN*)	111
Number of cycles (*MCN*)	279
Limit (*limit*)	3
DAX 100 index with 85 assets	
Number of food sources (*SN*)	185
Number of cycles (*MCN*)	459
Limit (*limit*)	3
FTSE 100 index with 89 assets	
Number of food sources (*SN*)	189
Number of cycles (*MCN*)	479
Limit (*limit*)	3
S&P 100 index with 98 assets	
Number of food sources (*SN*)	198
Dumber of cycles (*MCN*)	494
Limit (*limit*)	3
Nikkei index with 225 assets	
Number of food sources (*SN*)	300
Dumber of cycles (*MCN*)	750
Limit (*limit*)	3

ABC-FS global parameters, FA search parameters, portfolio problem parameters and constraint-handling parameters. Parameters are summarized in the Table 1.

6.2 Experimental results and comparative analysis

In the experimental section, we present the results obtained when searching the general efficient frontier that provides the solution of the problem formulated in Eqs. (21-24). The test data were obtained from http://people.brunel.ac.uk/ mastjjb/jeb/orlib/portinfo.html.

These data refer to the weekly stock prices from March 1992 to September 1997 for the indexes: the Hong Kong Hang Seng with 31 assets, the German Dax 100 with 85 assets, the British FTSE 100 with 89 assets, the US S&P 100 with 98 assets, and the Japanese Nikkei with 225 assets.

As mentioned in the previous subsection, *SN*, *MCN* and *limit* parameters depend on the problem size N (number of assets). Exact values used in tests are given in the Table 2. Since those parameters are integers, formula result is rounded to the closest integer value.

All tests were performed on Intel CoreTM i7 4770K processor @4GHz with 8GB of RAM memory, Windows 7 x64 Ultimate 64 operating system and Visual Studio 2010 .NET 4.0 Framework.

When sets of Pareto optimal portfolios obtained with ABC-FS metaheuristic are taken, heuristic efficient frontier can be traced. In this paper, we compare the standard efficient frontiers of the five real-world benchmark sets mentioned above with the heuristic efficient frontier for the same data set. For comparison of standard and heuristic efficiency frontier, we use mean Euclidean distance, variance of return error and mean return error as in [28]. We also give the execution time of ABC-FS for each benchmark on our computer platform.

For calculation purposes of mean Euclidean distance, let the pair $(v_i^s, r_i^s) = (i = 1,2,3,...,2000)$ denotes the variance and mean return of the point in the standard efficient frontier, and the pair $(v_j^h, r_j^h) = (i = 1,2,3,...,\xi)$ represents the variance and mean return of the point in the heuristic efficient frontier. Then, the closest standard efficiency frontier point to the heuristic efficiency frontier point, denoted as $(v_{i,j}^s, r_{i,j}^s)$ is calculated using Euclidean distance by:

$$i_j = arg\,min_{i=1,2,3,...2000}\,(\sqrt{(v_i^s - v_j^h)^2 + (r_i^s - r_j^h)^2}, \qquad j = 1,2,3,...\xi \tag{40}$$

According to Eq. (40), mean Euclidean distance is defined as:

$$\frac{(\sum_{j=1}^{\xi}\sqrt{(v_{i,j}^s - v_j^h)^2 - (r_{i,j}^s - r_j^h)^2})}{\xi} \tag{41}$$

Table 3 Experimental results for five benchmark indexes

Index	N	Performance indicators	GA	TS	SA	PSO	ABC-FS
Hang Seng	31	Mean Euclidean distance	0.0040	0.0040	0.0040	0.0049	**0.0004**
		Variance of return error (%)	1.6441	1.6578	1.6628	2.2421	**1.3952**
		Mean return error (%)	0.6072	0.6107	0.6238	0.7427	**0.5289**
		Execution time	18	9	10	34	12
DAX 100	85	Mean Euclidean distance	0.0076	0.0082	0.0078	0.0090	**0.0009**
		Variance of return error (%)	7.2180	9.0309	8.5485	**6.8588**	7.2649
		Mean return error (%)	**1.2791**	1.9078	1.2817	1.5885	1.35229
		Execution time	99	42	52	179	62
FTSE 100	89	Mean Euclidean distance	0.0020	0.0021	0.0021	0.0022	**0.0003**
		Variance of return error (%)	2.8660	4.0123	3.8205	3.0596	**2.6721**
		Mean return error (%)	0.3277	0.3298	0.3304	0.3640	**0.31872**
		Execution time	106	42	55	190	76
S%P 100	98	Mean Euclidean distance	0.0041	0.0041	0.0041	0.0052	**0.0001**
		Variance of return error (%)	**3.4802**	5.7139	5.4247	3.9136	3.7598
		Mean return error (%)	1.2258	**0.7125**	0.8416	1.4040	0.95292
		Execution time	126	51	66	214	125
Nikkei	225	Mean Euclidean distance	0.0093	0.0010	0.0010	0.0019	**0.0000**
		Variance of return error (%)	1.2056	1.2431	**1.2017**	2.4274	1.69823
		Mean return error (%)	5.3266	0.4207	**0.4126**	0.7997	0.67192
		Execution time	742	234	286	919	329

Besides mean Euclidean distance, we employed two other measures, variance of return error and mean return error. Variance of return error is defined as:

$$(\sum_{j=1}^{\xi} 100|v_{i,j}^{s} - v_{j}^{h}|/v_{j}^{h})\frac{1}{\xi} \tag{42}$$

Mean return error is calculated as:

$$(\sum_{j=1}^{\xi} 100|r_{i,j}^{s} - r_{j}^{h}|/r_{j}^{h})\frac{1}{\xi} \tag{43}$$

We compared ABC-FS with tabu search (TS), genetic algorithm (GA), simulated annealing (SA) from [23], and PSO from [28] for the same data set. We compared mean Euclidean distance, variance of return error and mean return error. We also give computational time for ABC-FS, but those results are incomparable with results for other metaheuristics because we used different computer platform. In [28] for testing purposes Pentium M 2.13 GHz computer with 1 GB RAM was used. In the results table, best obtained results of all five heuristics are printed bold.

The experimental results presented in Table 3 prove that none of the four algorithms which we took for comparisons has distinct advantages. However, it can be seen that on average, ABC-FS is better approach than other four metaheuristics for tackling CCMV portfolio problem.

ABC-FS obtains better (smaller) mean Euclidean distance for all five benchmark sets. In *HangSeng* and *FTSE*100 benchmarks, ABC-FS is better than all four algorithms in all three indicators, mean Euclidean distance, variance of return error and mean return error. For used benchmarks, ABC-FS was able to approximate the standard efficient frontier with the smallest mean return and variance of return error, and under the same risk values, ABC-FS's obtained portfolio return is higher than in the case of other algorithms.

Second best algorithms shown in Table 3 are SA which obtains best variance of return and mean return error in *Nikkei*225 test, and GA that shows satisfying results in *DAX*100 and *S*%*P*100 benchmarks. TS generated best value for mean return error in S%P 100 index, while PSO obtained best variance of return error in *DAX*100 test.

From the presented analysis it can be concluded that the our approach obtained results in CCMV portfolio optimization problem that can be more valuable for the investors. ABC-FS's results are more accurate and the generated investment strategy is able to more efficiently diversify the risk of the portfolio.

7 Conclusions

In this paper we present enhancement of the ABC algorithm by hybridization with the FA metaheuristic for cardinality constrained MV (CCMV) portfolio optimization problem. We adopted FA search procedure for the employed bees. Original ABC suffers from slow convergence and unbalanced trade-off between exploitation and exploration. By introducing FA search into ABC, we overcame this deficiency.

According to the test results and comparative analysis we conclude that in overall, ABC-FS is better than all state-of-the-art algorithms taken for the purpose of comparative analysis. Euclidean distance is smaller in all five benchmarks, and in $HangSeng$ and $FTSE100$ tests, ABC-FS outperformed all four metaheuristics in terms of all three indicators - mean Euclidean distance, mean variance of return and mean return error.

Future research may include application of here proposed ABC-FS algorithm to other portfolio optimization models and formulations with different constraints. Also, additional modifications of the ABC algorithm can be investigated for possible further improvement of results.

References

[1] Mezura-Montes (editor), E. *Constraint-Handling in Evolutionary Optimization*, vol. 198 of *Studies in Computational Intelligence* (Springer-Verlag, 2009).

[2] Tang, K., Yang, J., Chen, H. & Gao, S. Improved genetic algorithm for nonlinear programming problems. *Journal of Systems Engineering and Electronics* **22**, 540–546 (2011).

[3] Dorigo, M. & Gambardella, L. M. Ant colonies for the travelling salesman problem. *Biosystems* **43**, 73–81 (1997).

[4] Gan, R., Guo, Q., Chang, H. & Yi, Y. Improved ant colony optimization algorithm for the traveling salesman problems. *Journal of Systems Engineering and Electronics* **21**, 329–333 (2010).

[5] Jovanovic, R. & Tuba, M. An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. *Applied Soft Computing* **11**, 5360–5366 (2011).

[6] Jovanovic, R. & Tuba, M. Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem. *Computer Science and Information Systems (ComSIS)* **10**, 133–149 (2013).

[7] Tuba, M. & Jovanovic, R. Improved ant colony optimization algorithm with pheromone correction strategy for the traveling salesman problem. *International Journal of Computers, Communications & Control* **8**, 477–485 (2013).

[8] Zhou, Y., Zhou, G. & Zhang, J. A hybrid glowworm swarm optimization algorithm for constrained engineering design problems. *Applied Mathematics & Information Sciences* **7**, 379–388 (2013).

[9] Dai, C., Chen, W., Song, Y. & Zhu, Y. Seeker optimization algorithm: a novel stochastic search algorithm for global numerical optimization. *Journal of Systems Engineering and Electronics* **21**, 300–311 (2010).

[10] Tuba, M., Brajevic, I. & Jovanovic, R. Hybrid seeker optimization algorithm for global optimization. *Applied Mathematics & Information Sciences* **7**, 867–875 (2013).

[11] Yang, X.-S. & Deb, S. Cuckoo search via levy flights. In *Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, 210–214 (2009).

[12] Yang, X.-S. & Deb, S. Engineering optimization by cuckoo search. *International Journal of Mathematical Modeling and Numerical Optimization* **1**, 330–343 (2010).

[13] Brajevic, I. & Tuba, M. Cuckoo search and firefly algorithm applied to multilevel image thresholding. In Yang, X.-S. (ed.) *Cuckoo Search and Firefly Algorithm: Theory and Applications*, vol. 516 of *Studies in Computational Intelligence*, 115–139 (Springer International Publishing, 2014).

[14] Tuba, M. Asymptotic behavior of the maximum entropy routing in computer networks. *Entropy* **15**, 361–371 (2013).

[15] Karaboga, D. An idea based on honey bee swarm for numerical optimization. *Technical Report - TR06* 1–10 (2005).

[16] Yang, X.-S. Firefly algorithms for multimodal optimization. *Stochastic Algorithms: Foundations and Applications, LNCS* **5792**, 169–178 (2009).

[17] Srinivas, N. & Deb, K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation* **2**, 221–248 (1994).

[18] Deb, K., Amrit, P., Agarwal, S. & Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* **6**, 182–197 (2002).

[19] Lin, D., Wang, S. & Yan, H. A multiobjective genetic algorithm for portfolio selection problem. *Proc. of the 5th Internationa Conference of on Optimization: Techniques and Applications - ICOTA 2001* 25–31 (2001).

[20] Streichert, F., Ulmer, H. & Zell, A. Comparing discrete and continuous genotypes on the constrained portfolio selection problem. *LNCS* **3103**, 1239–1250 (2004).

[21] Tsao, C.-Y. & Liu, C.-K. Incorporating value-at-risk in portfolio selection: An evolutionary approach. In *Proc. of the Joint Conference on Information Sciences - JCIS 2006*, 115–123 (2006).

[22] Eddelbuttel, D. A hybrid genetic algorithm for passive management. *Second conference on computing in economics and Finance* 67–75 (1996).

[23] Chang, T., Meade, N., Beasley, J. & Sharaiha, Y. Heuristics for cardinality constrained portfolio optimization. *Computers and Operations Research* **27**, 1271–1302 (2000).

[24] Soleimani, H. & Hamid Reza Golmakani, M. H. S. Markowitz-based portfolio selection with minimum transaction lots, cardinality constraints and regarding sector capitalization using genetic algorithm. *Expert Systems with Applications* **36**, 5058–5063 (2009).

[25] Fernandez, A. & Gamez, S. Portfolio selection using neural networks. *Computers & Operations Research* **34**, 1177–1191 (2007).

[26] Deng, G.-F. & Lin, W.-T. Ant colony optimization for markowitz mean-variance portfolio model. *Swarm, Evolutionary, and Memetic Computing LNCS* **6466**, 238–245 (2010).

[27] Haqiqi, K. F. & Kazemi, T. Ant colony optimization approach to portfolio optimization - a lingo companion. *International Journal of Trade, Economics and Finance* **3**, 148–153 (2012).

[28] Cura, T. Particle swarm optimization approach to portfolio optimization. *Nonlinear Analysis: RealWorld Applications* **10**, 2396–2406 (2008).

[29] Zhu, H., Wang, Y., Wang, K. & Chen, Y. Particle swarm optimization (PSO) for the constrained portfolio optimization problem. *Expert Systems with Applications* **38**, 10161–10169 (2011).

[30] Wang, Z., Liu, S. & Kong, X. Artificial bee colony algorithm for portfolio optimization problems. *International Journal of Advancements in Computing Technology* **4**, 8–16 (2012).

[31] Wang, Z., Ouyang, R. & Kong, X. A hybrid artificial bee colony for portfolio optimization. *Journal of Theoretical and Applied Information Technology* **49**, 94–100 (2013).

[32] Tuba, M., Bacanin, N. & Pelevic, B. Artificial bee colony algorithm for portfolio optimization problems. *International Journal of Mathematical Models And Methods In Applied Sciences* **7**, 888–896 (2013).

[33] Chengli, Z. & Yan, C. Coherent risk measure based on relative entropy. *Applied Mathematics & Information Sciences* **6**, 233–238 (2012).

[34] Eiteman, D. K., Stonehill, A. I. & Moffett, M. H. *Multinational Business in Finance 13th edition*. Pearson series in France (Pearson, 2013).

[35] di Tollo, G. & Roli, A. Metaheuristics for the portfolio selection problem. *International Journal of Operations Research* **15**, 13–35 (2008).

[36] Markowitz, H. Portfolio selection. *The Journal of Finance* **7**, 77–91 (1952).

[37] Anagnostopoulos, K. P. & Mamanis, G. Multiobjective evolutionary algorithms for complex portfolio optimization problems. *Computational Management Science* **8**, 259–279 (2011).

[38] Golmakani, H. R. & Fazel, M. Constrained portfolio selection using particle swarm optimization. *Expert Systems with Applications* **38**, 8327–8335 (2011).

[39] Sharpe, W. F. Mutual fund performance. *The Journal of Business* **39**, 119–138 (1966).

[40] Corazza, M. & Favaretto, D. On the existence of solutions to the quadratic mixed-integer mean-variance portfolio selection problem. *European Journal of Operational Research* **176**, 1947–1960 (2007).

[41] Brankea, J., Scheckenbacha, B., Steina, M., Deb, K. & Schmecka, H. Portfolio optimization with an envelope-based multi-objective evolutionary algorithm. *European Journal of Operational Research* **199**, 684–693 (2009).

[42] Brajevic, I. & Tuba, M. An upgraded artificial bee colony algorithm (ABC) for constrained optimization problems. *Journal of Intelligent Manufacturing* **24**, 729–740 (2013).

[43] Bacanin, N. & Tuba, M. Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators. *Studies in Informatics and Control* **21**, 137–146 (2012).

[44] Subotic, M. & Tuba, M. Parallelized multiple swarm artificial bee colony algorithm (MS-ABC) for global optimization. *Studies in Informatics and Control* **23**, 117–126 (2014).

[45] Deb, K. An efficient constraint-handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering* **186**, 311–338 (2000).

[46] Deb, K. *Optimization for Engineering Design, Algorithms and Examples* (Prentice-Hall, 2005).

[47] Mezura-Montes, E. & Coello-Coello, C. A. Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation* **1**, 173–194 (2011).

Milan Tuba is Professor of Computer Science and Provost for Mathematical, Natural and Technical sciences at Megatrend University of Belgrade. He received B. S. in Math., M. S. in Math., M. S., M. Ph., Ph. D. in Computer Science from University of Belgrade and New York University. From 1983 to 1994 he was in the U.S.A. at Vanderbilt University in Nashville, Courant Institute of Mathematical Sciences, New York University and Cooper Union Graduate School of Engineering, New York. From 1994 he was Professor of Computer Science and Director of Computer Center at University of Belgrade, and from 2004 also a Dean of the College of Computer Science, Megatrend University Belgrade. His research interest includes mathematical, queuing theory and heuristic optimizations applied to computer networks, image processing and combinatorial problems. He has been an invited speaker of number of conferences and member of the editorial board or scientific committee of number of scientific journals and conferences and has published more than 150 scientific papers.

Nebojsa Bacanin was born in 1983. He is currently a Ph.D. candidate at the Department of Computer Science, University of Belgrade and teaching and research assistant at Faculty of Computer Science, Megatrend University, Belgrade. He is the coauthor of more than twenty scientific papers. His research interests include nature inspired optimization algorithms, data compression and numeric simulation.