Computers & Industrial Engineering 90 (2015) 339–351

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

# A GRASP based solution approach to solve cardinality constrained portfolio optimization problems ☆

Adil Baykasoğlu *, Mualla Gonca Yunusoglu, F. Burcin Özsoydan

*Dokuz Eylül University, Faculty of Engineering, Department of Industrial Engineering, Izmir, Turkey*

ARTICLE INFO

*Article history:*
Received 14 April 2015
Received in revised form 1 October 2015
Accepted 14 October 2015
Available online 20 October 2015

*Keywords:*
Portfolio optimization
Meta-heuristics
GRASP
Quadratic programming

ABSTRACT

In the current work, a solution methodology which combines a meta-heuristic algorithm with an exact solution approach is presented to solve cardinality constrained portfolio optimization (CCPO) problem. The proposed method is comprised of two levels, namely, stock selection and proportion determination. In stock selection level, a greedy randomized adaptive search procedure (GRASP) is developed. Once the stocks are selected the problem reduces to a quadratic programming problem. As GRASP ensures cardinality constraints by selecting predetermined number of stocks and quadratic programming model ensures the remaining problem constraints, no further constraint handling procedures are required. On the other hand, as the problem is decomposed into two sub-problems, total computational burden on the algorithm is considerably reduced. Furthermore, the performance of the proposed algorithm is evaluated by using benchmark data sets available in the OR Library. Computational results reveal that the proposed algorithm is competitive with the state of the art algorithms in the related literature.



## 1. Introduction

Portfolio optimization problem is focused on investing a predefined amount of capital in a number of assets in an effective way. The problem was expressed quantitatively by Markowitz (1952). Markowitz (1952) formulated the problem as a bi-objective optimization problem that maximizes expected return of the portfolio and minimizes risk (variance of the return) of the portfolio which means that the proposed mean–variance optimization model is based on the tradeoff between risk and return. However, direct application of the proposed model is not of much practical use mainly due to its simplified assumptions. From the practical point of view, real-world investors commonly face restrictions such as cardinality and floor–ceiling constraints.

The cardinality constraint imposes a limit on the number of assets in the portfolio either to simplify the management of the portfolio or to reduce transaction costs. The floor–ceiling constraint restricts the proportion of each asset in the portfolio to lie between the lower and upper bounds in order to avoid very small or large and unrealistic holdings (Lwin & Qu, 2013).

However, by inclusion of the practical constraints, particularly cardinality constraints, finding the solution of the problem becomes harder using exact optimization algorithms. Thus, application of meta-heuristic approaches to constrained portfolio optimization problems has gained attention of the researchers recently. Firstly, Chang, Meade, Beasley, and Sharaiha (2000) illustrate the differences arising in the shape of this efficient frontier in the presence of cardinality constraints. They utilized genetic algorithm (GA), tabu search (TS) and simulated annealing (SA) to find the cardinality constrained efficient frontier and presented computational results for the benchmark data sets available at the OR-Library (Beasley, 1990).

Recently, Cura (2009) applied a particle swarm optimization (PSO) to solve cardinality constrained portfolio optimization (CCPO) problem. The results are compared to the results of GA, TS and SA implemented in the same study using the benchmark data set of (Beasley, 1990). In the studies of Anagnostopoulos and Mamanis (2009) the non-dominating sorting genetic algorithm (NSGA) is used for solving the CCPO problem. The authors applied their algorithm to Athens Stock Exchange and compared the results to the results obtained by a GA. In a distinguishing work of Anagnostopoulos and Mamanis (2010), the portfolio optimization problem is formulated as a tri-objective optimization problem so as to find tradeoffs between risk, return and the number of securities in the portfolio. They use NSGA-II, Pareto envelope-based selection algorithm (PESA) and strength Pareto evolutionary algorithm 2 (SPEA2) and provided a performance comparison among the algorithms. In another work, Gupta, Mehlawat, and Mittal

☆ This manuscript was processed by Area Editor Elghazali Talbi.

* Corresponding author.
*E-mail address:* adil.baykasoglu@deu.edu.tr (A. Baykasoğlu).

http://dx.doi.org/10.1016/j.cie.2015.10.009
0360-8352/

(2011) presented an integrated approach for portfolio selection in a multi-criteria decision making framework. Firstly, they used support vector machines for classifying financial assets in three predefined classes, based on their performance on some key financial criteria. Next, they employed GA to solve a portfolio selection problem in the respective classes incorporating investor preferences. Anagnostopoulos and Mamanis (2011) utilized five multi-objective evolutionary algorithms, namely, niched Pareto genetic algorithm 2 (NPGA2), NSGA-II, PESA, SPEA2, and e-multi-objective evolutionary algorithm (e-MOEA) to solve the CCPO problem. In the same year, Woodside-Oriakhi, Lucas, and Beasley (2011) examined the performances of GA, TS and SA approaches to find the cardinality constrained efficient frontier. They compared the results with the results of Chang et al. (2000). Deng, Lin, and Lo (2012) used PSO to solve the CCPO problem. They compared the results with the results of GA, TS and SA using the benchmark data sets available in the OR Library (Beasley, 1990). Another study using PSO is presented by Sadigh, Mokhtari, Iranpoor, and Ghomi (2012). They proposed a hybrid approach that combines Hopfield neural network (HNN) and PSO to solve the CCPO problem. Lwin and Qu (2013) considered the extended mean–variance model with practical trading constraints including the cardinality, floor and ceiling constraints. They proposed a hybrid algorithm combining population based incremental learning and differential evolution (DE).

In the current study, a meta-heuristic algorithm (GRASP-QUAD) is proposed to solve the CCPO problem. Unlike the common solution approaches reported in the literature, the problem is handled in two levels simultaneously, namely, "stock selection" and "proportion determination". In the stock selection level, GRASP (Feo & Resende, 1995) is utilized to find desirable stocks to invest. As GRASP is a constructive algorithm, all solutions are constructed via GRASP by satisfying the cardinality constraints. Therefore, an explicit procedure is not required to handle the cardinality constraints. As a result of the stock selection level, CCPO is reduced to a quadratic programming problem where the aim is to find the proportions for the desirable stocks in the portfolio. Hence, in the proportion determination level, proportions for the selected stocks are determined by a quadratic programming model which satisfies the rest of the problem constraints itself.

The reminder of this paper is organized as follows. In Section 2, the portfolio optimization problem considered in this study is explained briefly. In Section 3, a detailed description of the proposed algorithm is provided. In Section 4, computational experiments and results are presented. Finally, concluding remarks are given in Section 5.

## 2. Portfolio optimization problem

The objective of the portfolio optimization problem is to select the stocks that will maximize the expected return and minimize the risk of the resulting portfolio. Markowitz (1952) quantifies portfolio risk for the first time and develops the standard portfolio optimization problem presented in the following.

$$\min \sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \sigma_{ij} \tag{1}$$

subject to

$$\sum_{i=1}^{N} x_i \mu_i = R^* \tag{2}$$

$$\sum_{i=1}^{N} x_i = 1, \tag{3}$$

$$0 \leqslant x_i \leqslant 1, \quad i = 1, \ldots, N. \tag{4}$$

where $N$ is the number of available stocks, $x_i$ is the proportion of stock $i$ in the portfolio, $\mu_i$ is the expected return of stock $i$, $\sigma_{ij}$ is the covariance between the returns of stocks $i$ and $j$, $R^*$ is the desired expected return. Herein, Eq. (1) minimizes the variance (risk) of the portfolio, Eq. (2) ensures the return of portfolio is equal to the desired return and Eq. (3) is the budget constraint imposing available budget to be invested.

To reflect the effect of investor's risk profile, an objective weighting parameter $\lambda \in [0, 1]$ is introduced to the standard model in the literature. With the objective weighting parameter, the standard model turns out to be:

$$\min \ \lambda \left[\sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \sigma_{ij}\right] - (1-\lambda)\left[\sum_{i=1}^{N} x_i \mu_i\right] \tag{5}$$

subject to

$$\sum_{i=1}^{N} x_i = 1, \tag{6}$$

$$0 \leqslant x_i \leqslant 1, \quad i = 1, \ldots, N. \tag{7}$$

when $\lambda$ is zero, the model maximizes the mean return of the portfolio, regardless of the variance (risk). In contrast, when $\lambda$ equals unity, the model minimizes the risk of the portfolio regardless of the mean return. Therefore, the sensitivity of the investor to the risk increases as $\lambda$ approaches unity, while it decreases as $\lambda$ approaches to zero. Each case with different $\lambda$ value would have a different objective function value, which is composed of mean return and variance. Tracing the mean return and variance intersections, a continuous curve called efficient frontier is obtained. Since every point on an efficient frontier curve indicates an optimal solution, the portfolio optimization problem is a multi-objective optimization problem (Cura, 2009). Fig. 1 illustrates standard efficient frontier derived for the Hang Seng benchmark problem available at the OR-Library (Beasley, 1990).

However, the standard mean–variance model has some limitations in addressing the requirements of the real world applications. Therefore, in this study, two additional constraint sets, namely, cardinality and floor–ceiling constraints, are added to the standard mean–variance model. As a result, the model turns out to be a mixed integer quadratic optimization model. The resulting CCPO model is presented in the following.

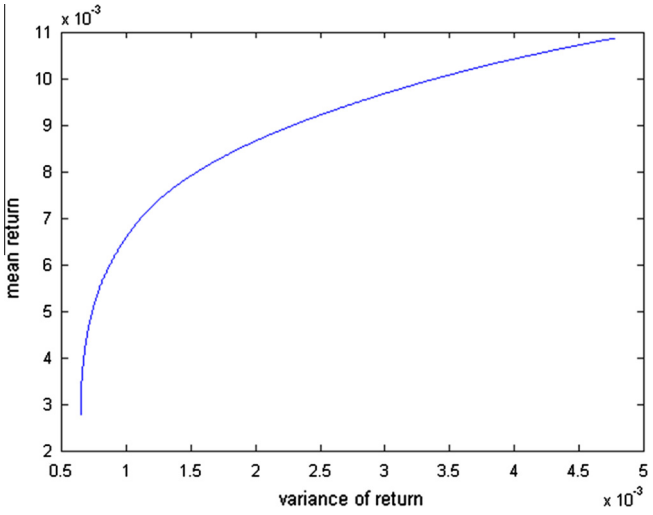



**Fig. 1.** Standard efficient frontier of Hang Seng benchmark problem.

$$\min \lambda \left[\sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \sigma_{ij}\right] - (1-\lambda)\left[\sum_{i=1}^{N} x_i \mu_i\right] \tag{8}$$

subject to

$$\sum_{i=1}^{N} x_i = 1, \tag{9}$$

$$\sum_{i=1}^{N} z_i = K, \tag{10}$$

$$\varepsilon_i z_i \leqslant x_i \leqslant \delta_i z_i, \quad i = 1,\ldots,N \tag{11}$$

$$0 \leqslant x_i \leqslant 1, \quad i = 1,\ldots,N \tag{12}$$

$$z_i \in \{0,1\}, \quad i = 1,\ldots,N. \tag{13}$$

Herein, the decision variable $z_i$ will be 1, if stock $i$ is included in the portfolio; otherwise it will be zero. Cardinality constraint (Eq. (10)) ensures that the portfolio consists of exactly $K$ stocks. Floor and ceiling constraints (Eq. (11)) limit proportion of each selected stock $i$ to the range $[\varepsilon_i, \delta_i]$. Finally Eqs. (12) and (13) impose restrictions on decision variables.

## 3. The proposed algorithm (GRASP-QUAD) for CCPO

Because of the binary variables, cardinality constraints and quadratic terms in CCPO problem, using exact solution approaches directly is neither effective nor efficient in the practice. Particularly, in the presence of binary variables, branching based solution techniques like branch and bound, branch and cut, branch and price or combinations of them may be used. In this case, the complexity of the problem and the required CPU time can exponentially increase as the size of the problem increases. Hence, a solution methodology, combining GRASP and quadratic programming is proposed to solve the present CCPO problem. The proposed algorithm actually hybridizes a meta-heuristic algorithm with an exact solution approach and basically it has two levels, namely, "stock selection" and "proportion determination". One of the main advantages of the proposed two-level approach is the reduction of the computational burden for solving the CCPO.

In stock selection level, GRASP is utilized to select the desirable stocks for the portfolio. In GRASP, two local search procedures, namely, *minor local search* (MiLS) and *major local search* (MaLS) are employed. As the number of decision variables is fixed to $K$ in stock selection level, the problem turns out to be a quadratic optimization problem. In the proportion determination level, the proportions for the stocks selected by GRASP are determined by a quadratic programming model and the resulting portfolio is obtained. The flowchart of the proposed algorithm is presented in Fig. 2.

### 3.1. Stock selection level

In this level, $K$ desirable stocks out of $N$ candidate stocks are selected for the portfolio using GRASP (Feo & Resende, 1995) which is a constructive and multi-start meta-heuristic algorithm consisting of two phases, namely, solution construction and local search phases. The solution construction phase builds desirable solution satisfying cardinality constraints (Eq. (10)). Once a feasible solution is obtained, its neighborhood is searched in order to improve the constructed solution (Resende & Ribeiro, 2010). The obvious advantage of the GRASP is that it saves the algorithm from exponential growth in computation time depending on the increase of the problem size. As another advantage, it constructs a solution involving $K$ out of $N$ candidate stocks which means that no additional procedure is needed to satisfy the cardinality constraints.

#### 3.1.1. Solution construction

In the current work, a solution which is constructed by GRASP is represented as a sorted list (tuple) of the selected stocks (the first rows of Fig. 3a–c). Initially, the list of selected stocks is empty. In what follows, the greedy values (desirability) of all candidate stocks are evaluated by using Eq. (14) proposed by Sadigh et al. (2012). Herein, $\lambda$ is the objective weighting parameter used in Eq. (8).

$$f_s = \frac{1+\lambda\left(\frac{\sum_{j=1}^{N}\sigma_{sj}}{N}\right)}{1+(1-\lambda)\mu_s} \quad s = 1,\ldots,N \tag{14}$$

A stock with smaller greedy value means a more desirable stock if whole data is comprised of only positive values. However, some benchmarking data may include both positive and negative return values which may yield to positive or negative greedy values. In this case, a direct sort of stocks according to their greedy values will be unreasonable. On the other hand, encouraging the selection of the stocks with only positive greedy values while discouraging the selection of the stocks with negative greedy rules may probably ends up with local optima. Therefore, the candidate stocks with positive and negative greedy values are both sorted in non-decreasing order but they are kept separately in such cases. The sorted stocks with negative greedy values are appended to the end of the ones with positive greedy values and whole candidate stock list is generated (Fig. 3).

Afterwards, a stock is randomly selected from a subset of this generated list. This subset is referred as restricted candidate list (RCL) containing a predetermined number ($RCL_{size}$) of sorted stocks. If $RCL_{size}$ is equal to the size of the whole list, then it becomes a random selection. In this case, there is a probability that search might not be able to converge to a qualified solution. On the other hand, if $RCL_{size}$ is kept relatively too small, then the selection turns out to be a more greedy procedure and only the stocks with the most desirable stocks are selected. In other words, a myopic selection is performed and the search might be stuck at local optima. In brief, it can be said that the aim of using RCL is to tune the selection pressure and to keep a good balance between randomness and greediness.

An example for sorting candidate stocks, generating whole list and RCL is depicted in Fig. 3a–c. Whole list is comprised of all candidate stocks, while RCL is a subset of whole list where stocks can be selected. RCL is highlighted in Fig. 3a–c and the stocks in RCL are denoted in italic.

Assume that there are 10 unselected stocks in the whole candidate list and $RCL_{size}$ values in Fig. 3a–c are set to 2, 6 and 10, respectively. GRASP behaves more greedily in Fig. 3a, because only the two most desirable stocks (5 and 7) are allowed to be selected in portfolio. It means that whenever a stock is to be selected, if $RCL_{size}$ is set to 2, a random stock is chosen among from 2 stocks listed in RCL. In Fig. 3b, RCL is kept relatively at a moderate level, where stocks with negative greedy values are also encouraged to be selected. On the other hand, in Fig. 3c, $RCL_{size}$ is equalized to the size of whole list which means that all candidate stocks have equal chances to be selected. This yields to a totally random search. In this study, a similar approach as given in Fig. 3b is used and the size of the RCL is fixed approximately to the half of the number of the stocks in the benchmark data. $RCL_{size}$ is not changed until the termination of algorithm.

It's worth stressing that each time only one stock is selected from RCL and it is appended to the end of the list of the selected stocks. The stocks which are not listed in RCL cannot be selected, while the stocks in RCL have equal chances to be selected. Subsequent to selection of a stock, the selected stock is removed from the candidate stock list. Selecting a stock constructs a part of the

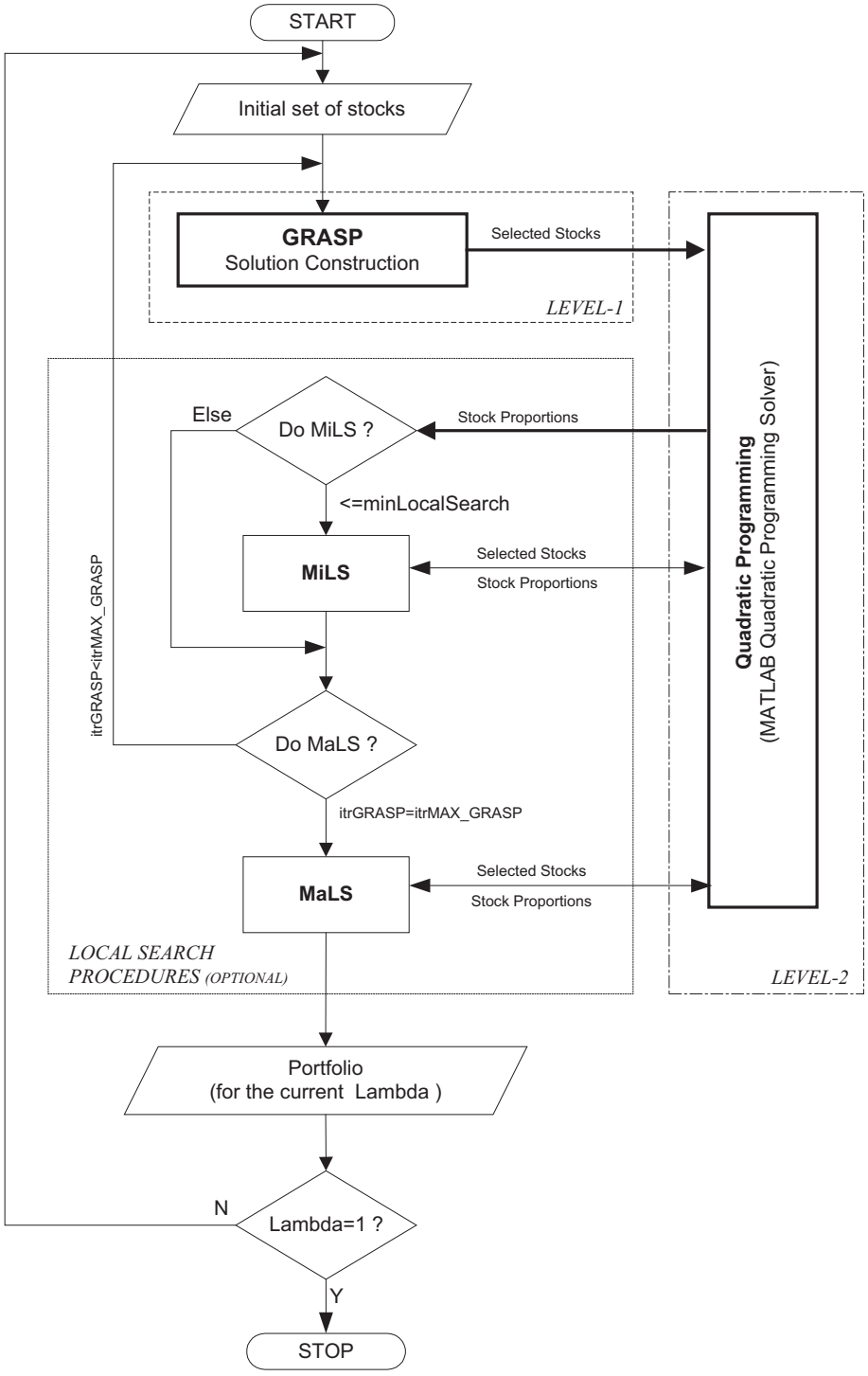



**Fig. 2.** The flowchart of GRASP-QUAD.

solution. Whole process (updating greedy values, sorting, updating whole list and RCL) is repeated until *K* stocks are placed in the portfolio and whole solution is constructed.

### 3.1.2. Local search

Generally, in GRASP, improvement heuristics are applied subsequent to solution construction phase. In the present study, two types local search techniques, namely, *minor local search* (*MiLS*) and *major local search* (*MaLS*), are used within the algorithm.

*3.1.2.1. Minor local search.* MiLS is performed depending on a probability (*minLocalSearch*) rather than applying it at each step of GRASP. The reason for relating MiLS with a probability is to achieve possible gains on computational effort. Additionally, trial tests demonstrated that applying MiLS at each step of GRASP did not improve the solution quality in comparison to applying it probabilistically.

MiLS is a simple and fast local search procedure which inserts an unselected stock in the place of a selected stock and checks whether an improvement in the objective value exists. Both selected and

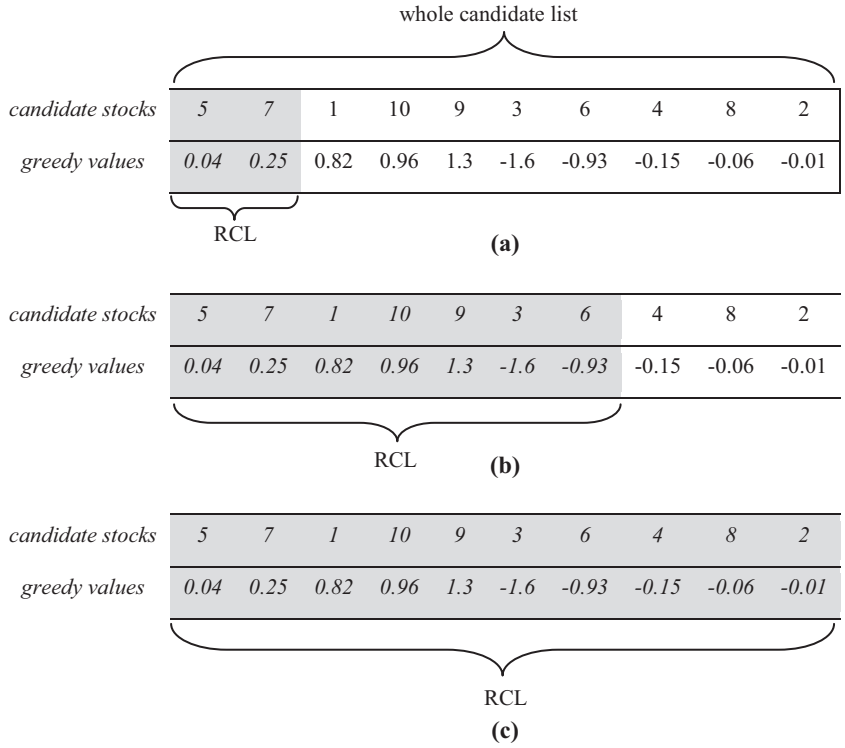



**Fig. 3.** (a) $RCL_{size}$ is set to a very small number. (b) $RCL_{size}$ is kept moderately. (c) $RCL_{size}$ is equal to size of whole candidate list.

unselected stocks are determined randomly. If the objective value is improved, newly generated solution and corresponding objective value are kept in a matrix and an array, respectively. This procedure is followed until a predetermined number of steps referred as *UB_minLocalSearch* is reached. In subsequent to achieving the final step of MiLS, if an improvement is detected, the best trial solution (with the minimum objective value) is driven from the matrix where enhancer solutions are kept throughout MiLS.

*3.1.2.2. Major local search.* In comparison to MiLS, MaLS is a more time consuming technique which increases the computational effort. Therefore, MaLS in only applied once to the best solution of each trade-off point found so far, at the final step of GRASP. The main difference between MaLS and MiLS is that the MiLS is run and terminated in single pass, where further improvements are not checked. However, in MaLS, the procedure is not limited to achieving a maximum number of iterations. As far as there exists possible improvements in the current solution, MaLS is run until no further improvements are possible. In MaLS, all swap combinations of the selected and unselected stocks are checked. The swaps, which are improving the current solution are kept in a matrix. Then, a random number is generated. If this random number if less than *randomFit*, which is the single parameter of MaLS, than a random solution is driven among from the matrix and current solution is replaced with this solution. Otherwise, the best solution in that matrix is driven and the current solution is replaced with the best improving solution. As a result, *randomFit* parameter tunes the tradeoff between a random fit and the best fit neighborhood selection. In order to prevent early convergence, random fit selections are also encouraged. Subsequent to selection of the best or random fit neighborhood, the same procedure is followed for the newly selected solution and new neighborhoods are generated using this new solution.

Functioning of the MaLS is very simple but effective as it can be seen from Fig. 4. One can easily recognize the difference for the dispersion and convergence of the trade-off points depicted in Fig. 4, where Hang Seng benchmark problem available at the OR Library is solved exactly with the same parameters but with only difference that neither MiLS nor MaLS is performed in Fig. 4a, only MiLS is performed in Fig. 4b and finally both MiLS and MaLS are performed in Fig. 4c. Herein, GRASP is run only for 100 iterations for each of them.

As a result of GRASP, $K$ desirable stocks are selected for investment. The proportions of the desirable stocks are determined at the second level of the proposed algorithm.

### 3.2. Proportion determination level

As $K$ desirable stocks are selected by GRASP previously, the CCPO problem reduces to a quadratic programming model as in the following.

$$\min \lambda \left[\sum_{i=1}^{K}\sum_{j=1}^{K} x_i x_j \sigma_{ij}\right] - (1-\lambda)\left[\sum_{i=1}^{K} x_i \mu_i\right] \tag{15}$$

subject to

$$\sum_{i=1}^{K} x_i = 1, \tag{16}$$

$$\varepsilon_i z_i \leqslant x_i \leqslant \delta_i z_i, \quad i = 1, \ldots, K \tag{17}$$

$$0 < x_i \leqslant 1, \quad i = 1, \ldots, K \tag{18}$$

The quadratic programming model presented in Eqs. (15)–(18) can easily be solved by using any exact solver supporting quadratic models. Because GRASP was coded by using MATLAB, quadratic programming solver of MATLAB was also used at the second level of the proposed algorithm. The most remarkable advantage of this preference is that no further effort for integrating the algorithms is required, since GRASP and quadratic programming solver use MATLAB platform simultaneously by sequence.

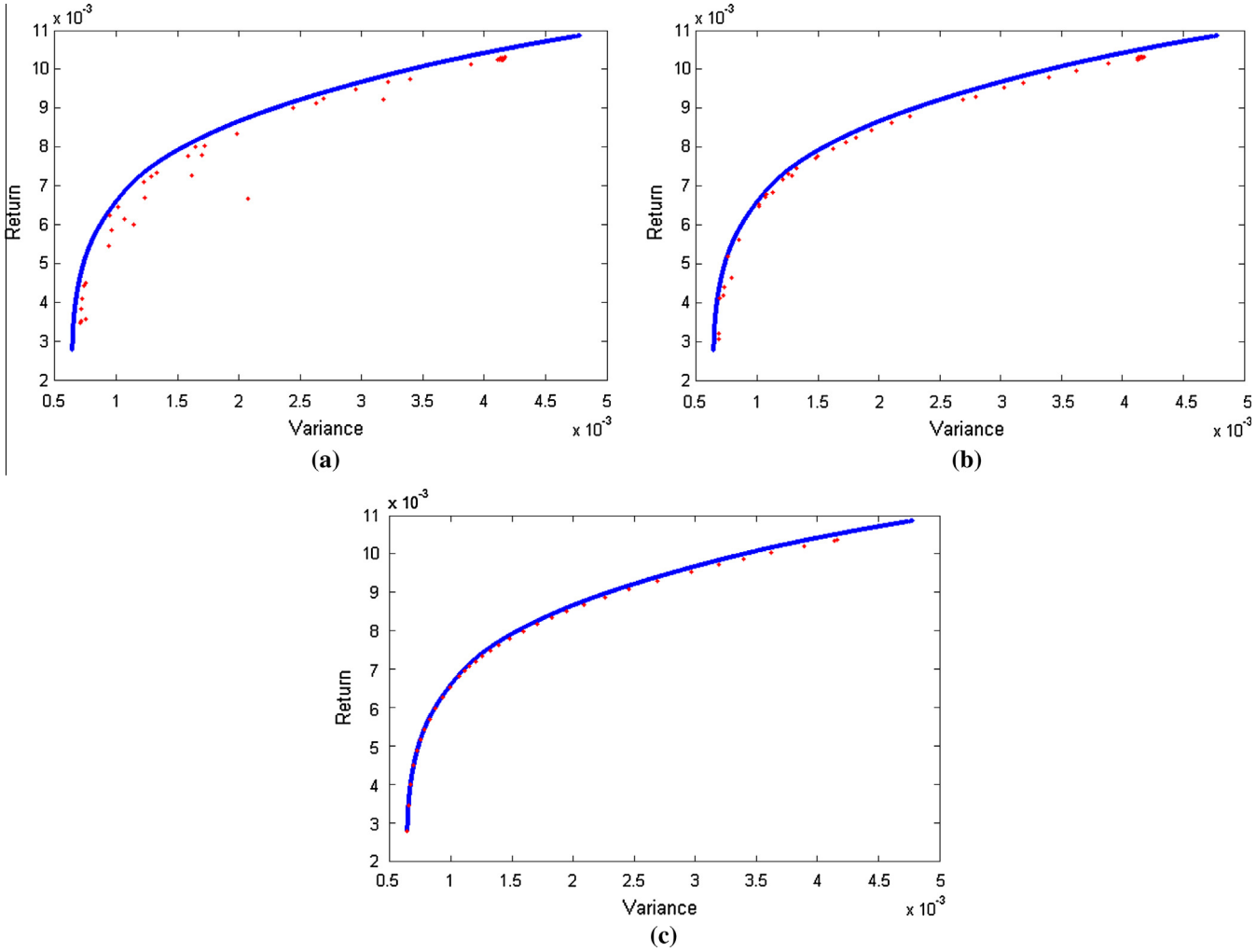



**Fig. 4.** Efficient frontiers obtained by using neither MiLS nor MaLS (a), by using MiLS (b) and by using both MiLS and MaLS (c).

There are several configuration parameters for quadratic programming solver of MATLAB. In the current work, default parameter setting of the solver is used except for *MaxIter* and *TolFun* parameters which were set to 5000 and 1e−16, respectively. The details of the solver can be found at http://www.mathworks.com/help/optim/ug/quadprog.html.

The pseudo code of the proposed algorithm is presented in Fig. 5.

## 4. Computational experiments

In this section, computational results generated by the proposed algorithm are presented and compared with the previously reported results of Fernández and Gómez (2007), Cura (2009), Woodside-Oriakhi et al. (2011), Sadigh et al. (2012), and Lwin and Qu (2013). The performance of the algorithms is evaluated on the benchmark data which is publicly available at the OR-Library (Beasley, 1990). Specifically, the data correspond to weekly prices between March 1992 and September 1997 from Hang Seng, DAX 100, FTSE 100, S&P 100 and Nikkei 225 indices. The number of stocks ($N$) for each data set is 31, 85, 89, 98 and 225, respectively. As in the studies of Fernández and Gómez (2007), Cura (2009), Woodside-Oriakhi et al. (2011), Sadigh et al. (2012) and Lwin and Qu (2013), the CCPO problem is solved by using the parameter set $K = 10$, $\varepsilon_i = 0.01$ $(i = 1, \ldots, N)$ and $\delta_i = 1$ $(i = 1, \ldots, N)$.

### 4.1. Performance measures

Chang et al. (2000) calculated 2000 distinct points on the standard efficient frontier of the unconstrained portfolio optimization model presented in Eqs. (1)–(4) by considering 2000 different return values ($R^*$) for each benchmark data (Beasley, 1990). For example, standard efficient frontier calculated for Hang Seng data set is illustrated in Fig. 1. However, the heuristic efficient frontiers (cardinality constrained efficient frontier) calculated in this study will be discontinuous and below of the standard efficient frontiers due to the practical constraints imposed to the model. Therefore, the aim of this study is to calculate a heuristic efficient frontier that is nearest to the standard efficient frontier. Thus, we trace out heuristic efficient frontier of each benchmark problem for 51 different $\lambda$ values by taking $\lambda = 0$ for the first run of the algorithm and incrementing $\lambda$ by 0.02 for the other runs (Fig. 5).

There are several performance measures for the CCPO problem published in the literature. Commonly used measures are adopted here. We compared heuristic efficient frontier obtained by the proposed algorithm with the corresponding standard efficient frontier by using the performance measures such that variance of the return error, mean return error, mean percentage error, median percentage error, minimum percentage error and maximum percentage error as also used by Fernández and Gómez (2007), Woodside-Oriakhi et al. (2011) and Lwin and Qu (2013). The calculation of those performance measures are as follows:

```
//Load test problem; configure input parameters;
for λ = 0 to 1
   while  itrGRASP ≤ itrMAX_GRASP
      S:=<>; //S: the list (tuple) of the selected stocks (it is initially empty)
      //stock selection (solution construction) phase
      for k=1 to K //K is the number of stocks to be selected
         evaluate greedy values f_s for each candidate stock s ∉ S// see Eq.14
         f⁺ := {f_s | f_s > 0, s ∉ S}; f⁻ := {f_s | f_s < 0, s ∉ S}
         sort f⁺ and f⁻ by non-decreasing order;
         if RCL_size ≤ n(f⁺)
            RCL := f⁺; // RCL: restricted candidate list;
         else
            l := RCL_size − n(f⁺);
            RCL := f⁺ ∪ {f_i | i ≤ l, ∀i, f_i ∈ f⁻};
         end if
         select a random stock r from RCL;
         S := S ∪ r; // randomly selected stock r is appended to the selected stock list
         S^c := M\r   //M is the list of all stocks
      end for
      [fVal weights]:=quadraticProgram(S);
      if rand ≤ minLocalSearch //minor local search, rand ∈ [0,1]
         improved:=0;
         for i=1 to UB_minLocalSarch
            r1 := ⌊rand × n(S)⌋+1;  r2 := ⌊rand × n(S^c)⌋+1; //rand ∈ [0,1]
            S^t := S; //S^t is trial solution vector, S^c is complementary vector
            S^t_r1 := S^c_r2 //random insertion of an unselected stock
            [fVal' weights']:=quadraticProgram(S^t);
            if fVal'<fVal
               improved:=1;
               keep the trial solution and its objective value;
            end if
         end for
         if improved==1
            update S with the best trial vector solution already kept in matrix and arrays;
         end if
      end if
      if itrGRASP==itrMAX_GRASP   //major local search
         gain=1;
         while gain=1
            index1:=<>; index2:= <>; index3:= <>; S^c := M\S^g; //S^g is the global best solution
            for i=1 to n(S^g)
               S^t := S^g;
               for j=1 to n(S^c)
                  S^t_i := S^c_j; [fVal' weights']:=quadraticProgram(S^t);
                  if fVal' < fVal^g
                     index1:=index1∪{i}; index2:=index2∪{j}; index3:=index3∪{fVal'}
                     end if
               end for
            end for
            if n(index1)==0
               gain:=0;
            else
               if rand ≤ randomFit  // rand ∈ [0,1]
                  random:=⌊rand × n(index1)⌋+1;  // rand ∈ [0,1]
                  a:=S^c_index2_random ;  b:=S^g_index1_random ;
                  S^g_index1_random :=a;  S^c_index2_random :=b;
                  [fVal^g weights^g]:=quadraticProgram(S^g);
               else
                  argmin_x{∀x ∈ index3};
                  a:= S^c_index2_x; b:= S^g_index1_x;
                  S^g_index1_x:=a;  S^c_index2_x:=b;
                  [fVal^g weights^g]:=quadraticProgram(S^g);
               end if
            end if
         end while
      end if
      itrGRASP:=itrGRASP+1; update global best if necessary;
   end while
   keep the best solution η^λ;
   λ := λ + 0.02;
end for
```

**Fig. 5.** Pseudo code for the proposed algorithm.

**Table 1**
Factor levels considered in experimental design.

| Level | $RCL_{size}$ | *minLocalSearch* | *UB_minLocalSearch* | *randomFit* |
|---|---|---|---|---|
| 1 | $0.4 \times N$ | 0.1 | 15 | 0.25 |
| 2 | $0.5 \times N$ | 0.2 | 25 | 0.50 |
| 3 | $0.6 \times N$ | 0.5 | 50 | 0.75 |

Let $(v_i^s, r_i^s)$, $(i = 1, \ldots, 2000)$ and $(v_j^h, r_j^h)$, $(j = 1, \ldots, 51)$ represent the variance and mean return of the $i$th point on the standard efficient frontier and of the $j$th point on the heuristic efficient frontier, respectively. The closest mean return coordinates bracketing $r_j^h$ are obtained as in Eqs. (19) and (20):

$$r_{m_j}^s = \min(r_i^s | r_i^s \geqslant r_j^h) \tag{19}$$

$$r_{n_j}^s = \max(r_i^s | r_i^s \leqslant r_j^h) \tag{20}$$

Let $\hat{v}_{i_j}^s$ (Eq. (21)) be the linearly interpolated variance coordinate of the point on the standard efficient frontier where $r = r_j^h$.

$$\hat{v}_{i_j}^s = v_{n_j}^s + (v_{m_j}^s - v_{n_j}^s)\frac{(r_j^h - r_{n_j}^s)}{(r_{m_j}^s - r_{n_j}^s)} \tag{21}$$

Let $\hat{r}_{i_j}^s$ (Eq. (22)) be the linearly interpolated variance coordinate of point on the standard efficient frontier where $v = v_j^h$.

$$\hat{r}_{i_j}^s = r_{n_j}^s + (r_{m_j}^s - r_{n_j}^s)\frac{(v_j^h - v_{n_j}^s)}{(v_{m_j}^s - v_{n_j}^s)} \tag{22}$$

where $v_{m_j}^s$ and $v_{n_j}^s$ are the variance of return coordinates bracketing $v_j^h$ calculated by the same method presented in Eqs. (19) and (20).

The percentage deviations on variance $(\varphi_j)$ and return $(\psi_j)$ for any heuristic point $(v_j^h, r_j^h)$ are calculated as in Eqs. (23) and (24).

$$\varphi_j = 100\left|\frac{v_j^h - \hat{v}_{i_j}^s}{\hat{v}_{i_j}^s}\right| \tag{23}$$

$$\psi_j = 100\left|\frac{r_j^h - \hat{r}_{i_j}^s}{\hat{r}_{i_j}^s}\right| \tag{24}$$

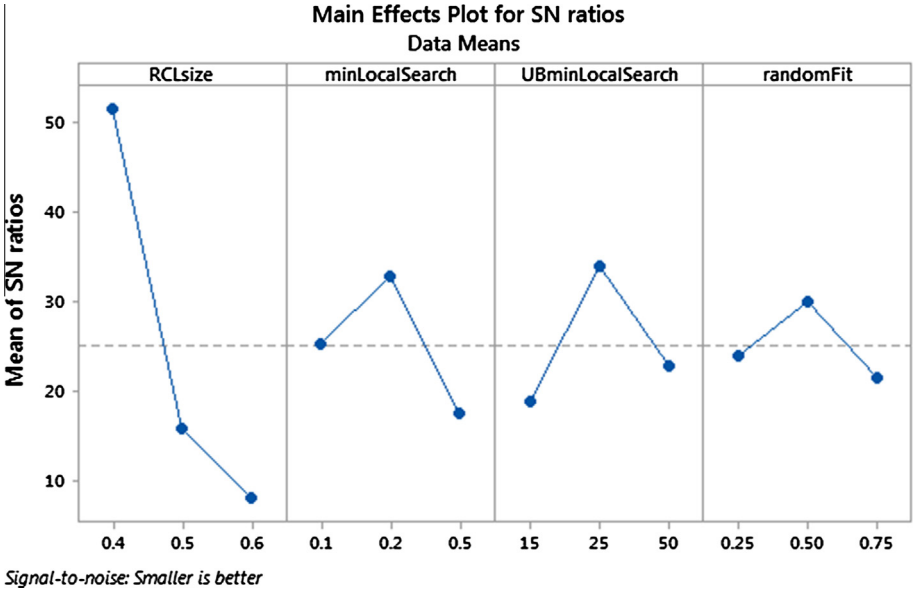



**Fig. 6.** Main effects for $S/N$ ratios.

**Table 2**
Computational results compared with the results reported by Fernández and Gómez (2007).

| Index | *N* | Percentage error | Fernández et al.-NN | GRASP-QUAD |
|---|---|---|---|---|
| Hang Seng | 31 | Variance of return | 1.2279 | 0.0567 |
| | | Mean return | 1.5304 | 0.0203 |
| | | Mean | 0.3751 | 0.0168 |
| DAX 100 | 85 | Variance of return | 2.3004 | 0.0865 |
| | | Mean return | 2.7545 | 0.0143 |
| | | Mean | 1.1326 | 0.0112 |
| FTSE 100 | 89 | Variance of return | 2.5449 | 0.0462 |
| | | Mean return | 3.1171 | 0.0118 |
| | | Mean | 1.2544 | 0.0104 |
| S&P 100 | 98 | Variance of return | 5.9020 | 2.0784 |
| | | Mean return | 4.7956 | 0.2086 |
| | | Mean | 2.7908 | 0.2082 |
| Nikkei | 225 | Variance of return | 0.7172 | 0.1401 |
| | | Mean return | 4.6097 | 0.0285 |
| | | Mean | 0.3578 | 0.0168 |

**Table 3**
Computational results compared with the results of Cura (2009) and Sadigh et al. (2012).

| Index | N | Performance measure | Cura-PSO | Sadigh, et al.-PSO-HNN | GRASP-QUAD |
|---|---|---|---|---|---|
| Hang Seng | 31 | Mean Euclidian distance | 0.0049 | 0.0001 | 0.0001 |
| | | Variance of return error (%) | 2.2421 | 2.5908 | 1.6400 |
| | | Mean return error (%) | 0.7427 | 0.7335 | 0.6060 |
| | | Time (s) | 34 | NA | 27 |
| DAX 100 | 85 | Mean Euclidian distance | 0.0090 | 0.0000 | 0.0001 |
| | | Variance of return error (%) | 6.8588 | 5.7585 | 6.7593 |
| | | Mean return error (%) | 1.5885 | 0.1466 | 1.2769 |
| | | Time (s) | 179 | NA | 86 |
| FTSE 100 | 89 | Mean Euclidian distance | 0.0022 | 0.0000 | 0.0000 |
| | | Variance of return error (%) | 3.0596 | 5.4141 | 2.435 |
| | | Mean return error (%) | 0.3640 | 0.3095 | 0.3245 |
| | | Time (s) | 190 | NA | 92 |
| S&P 100 | 98 | Mean Euclidian distance | 0.0052 | 0.0000 | 0.0001 |
| | | Variance of return error (%) | 3.9136 | 5.1456 | 2.5211 |
| | | Mean return error (%) | 1.4040 | 0.2925 | 0.9063 |
| | | Time (s) | 214 | NA | 96 |
| Nikkei | 225 | Mean Euclidian distance | 0.0019 | 0.0000 | 0.0000 |
| | | Variance of return error (%) | 2.4274 | 4.7779 | 0.8359 |
| | | Mean return error (%) | 0.7997 | 0.7040 | 0.4184 |
| | | Time (s) | 919 | NA | 409 |

**Table 4**
Computational results compared with the results reported by Woodside-Oriakhi et al. (2011) and Lwin and Qu (2013).

| Index | N | Percentage error | Woodside-Oriakhi et al.-GA | Woodside-Oriakhi et al.-TS | Woodside-Oriakhi et al.-SA | Lwin et al.-PBILDE | GRASP-QUAD |
|---|---|---|---|---|---|---|---|
| Hang Seng | 31 | Mean | 0.8501 | 0.8234 | 1.0589 | 0.6196 | 0.0107 |
| | | Median | 0.5873 | 0.3949 | 0.5355 | 0.4712 | 0.0064 |
| | | Minimum | 0.0036 | 0.0068 | 0.0349 | 0.2816 | 0.0000 |
| | | Maximum | 2.9034 | 4.6096 | 4.6397 | 0.6768 | 0.0366 |
| | | Time (s) | 76 | 85 | 99 | NA | 62 |
| DAX 100 | 85 | Mean | 0.7740 | 0.7190 | 1.0267 | 1.5433 | 0.0140 |
| | | Median | 0.2400 | 0.4298 | 0.8682 | 1.0986 | 0.0127 |
| | | Minimum | 0.0000 | 0.0149 | 0.0278 | 0.7537 | 0.0005 |
| | | Maximum | 4.6811 | 2.7770 | 4.4123 | 1.6804 | 0.0485 |
| | | Time (s) | 74 | 113 | 293 | NA | 139 |
| FTSE 100 | 89 | Mean | 0.1620 | 0.3930 | 0.8952 | 0.8234 | 0.0124 |
| | | Median | 0.0820 | 0.2061 | 0.3944 | 0.5134 | 0.0109 |
| | | Minimum | 0.0000 | 0.0019 | 0.0230 | 0.4359 | 0.0004 |
| | | Maximum | 0.7210 | 3.4570 | 10.2029 | 0.8695 | 0.0407 |
| | | Time (s) | 95 | 232 | 286 | NA | 143 |
| S&P 100 | 98 | Mean | 0.2922 | 1.0358 | 3.0952 | 1.3902 | 0.2775 |
| | | Median | 0.1809 | 1.0248 | 2.1064 | 0.7303 | 0.0158 |
| | | Minimum | 0.0007 | 0.0407 | 0.8658 | 0.4816 | 0.0000 |
| | | Maximum | 1.6295 | 3.0061 | 8.6652 | 1.5726 | 9.8784 |
| | | Time (s) | 100 | 222 | 371 | NA | 172 |
| Nikkei | 225 | Mean | 0.3353 | 0.7838 | 1.1193 | 0.3996 | 0.0063 |
| | | Median | 0.3040 | 0.6526 | 0.6877 | 0.4619 | 0.0029 |
| | | Minimum | 0.0180 | 0.0085 | 0.0113 | 0.3739 | 0.0000 |
| | | Maximum | 1.0557 | 2.6082 | 3.9678 | 0.4965 | 0.0279 |
| | | Time (s) | 104 | 414 | 604 | NA | 438 |
| Average over all instances | – | Mean | 0.4827 | 0.7510 | 1.4391 | 0.9552 | 0.0642 |
| | | Median | 0.2788 | 0.5416 | 0.9184 | 0.6550 | 0.0097 |
| | | Minimum | 0.0045 | 0.0146 | 0.1926 | 0.4653 | 0.0002 |
| | | Maximum | 2.1981 | 3.2916 | 6.3776 | 1.0591 | 2.0064 |
| | | Time (s) | 90 | 213 | 331 | NA | 191 |

To sum up, variance of return error and mean return error are calculated as in Eqs. (25) and (26), respectively:

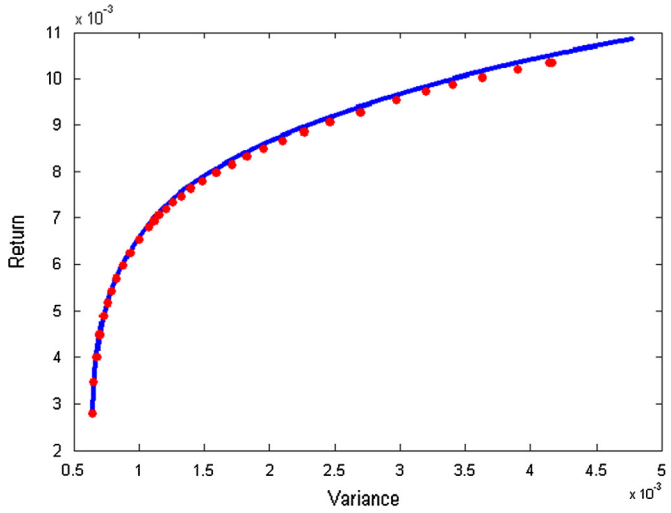
$$\textit{Variance of return error} = \sum_{j=1}^{51} \varphi_j / 51 \tag{25}$$

$$\textit{Mean return error} = \sum_{j=1}^{51} \psi_j / 51 \tag{26}$$

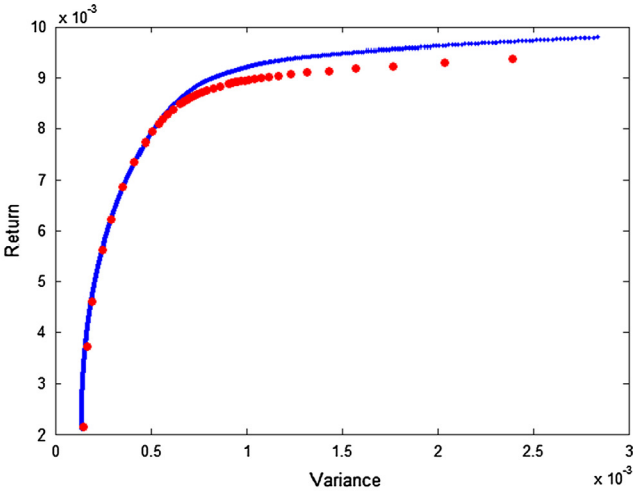
To calculate the mean percentage error, standard deviation of return error is used instead of variance of return error.

$$\omega_j = 100 \left| \frac{\sqrt{v_j^h} - \sqrt{\hat{v}_{i_j}^s}}{\sqrt{\hat{v}_{i_j}^s}} \right| \tag{27}$$

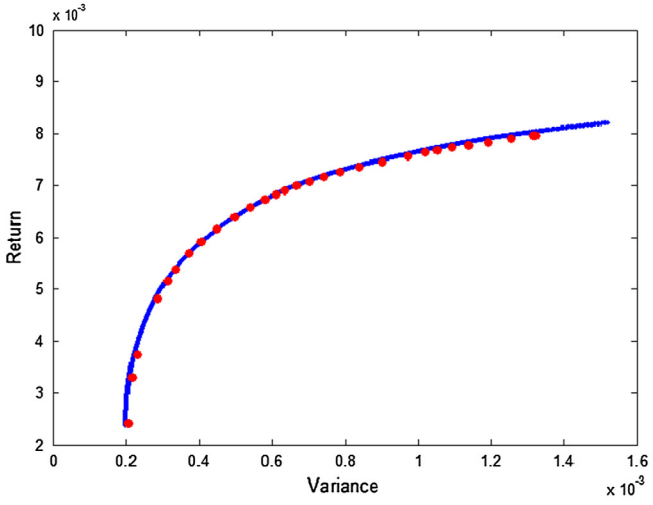
$$\textit{Mean percentage error} = \sum_{j=1}^{51} [\min(\omega_j, \psi_j)/51] \tag{28}$$



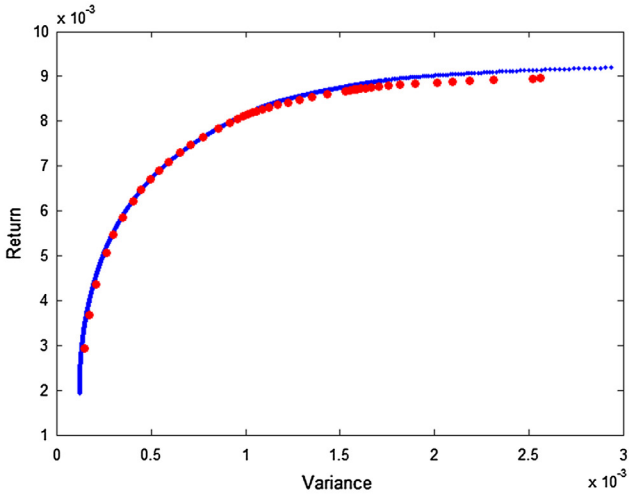

Hang Seng



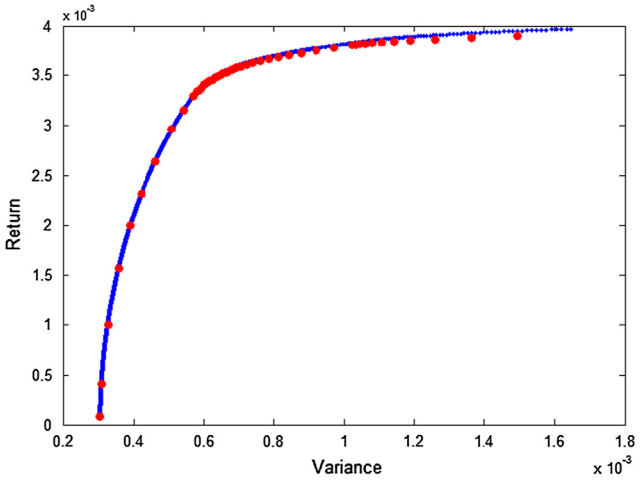

DAX 100




FTSE 100




S&P 100




Nikkei

**Fig. 7.** Standard and heuristic efficient frontiers obtained by GRASP-QUAD.

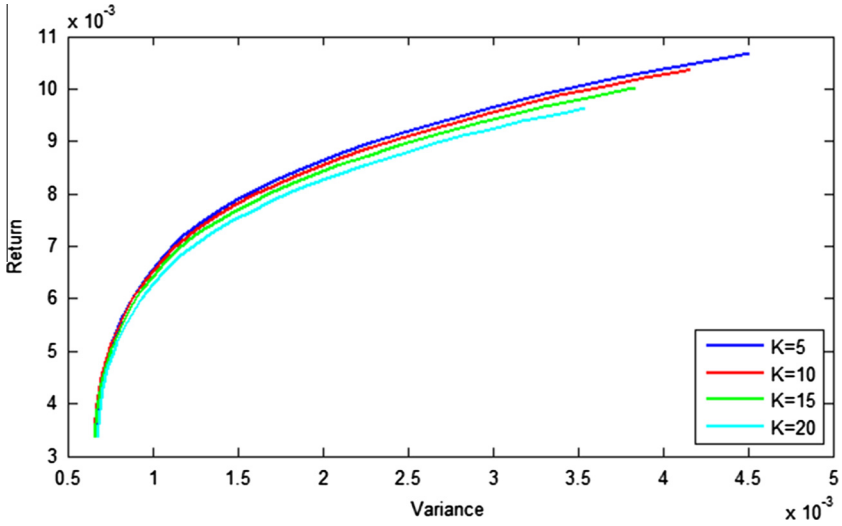



**Fig. 8.** Heuristic efficient frontiers for Hang Seng obtained for different cardinality values.

On the other hand, Cura (2009) and Sadigh et al. (2012) compared heuristic efficient frontier with standard heuristic frontier by using mean Euclidian distance, variance of return error and mean return error.

However, they calculated variance of the return error and mean return error in a different way from previously stated authors. Let $(v_{i_j}^s, r_{i_j}^s)$ be the closest standard point to the heuristic point $(v_j^h, r_j^h)$. Mean Euclidian distance, variance of return error and mean return error are calculated as follows:

$$\textit{mean Euclidian distance} = \left(\sum_{j=1}^{51}\sqrt{(v_{i_j}^s - v_j^h)^2 + (r_{i_j}^s - r_j^h)^2}\right) \times \frac{1}{51} \quad (29)$$

$$\textit{variance of return error} = \left(\sum_{j=1}^{51} 100|v_{i_j}^s - v_j^h|/v_j^h\right) \times \frac{1}{51} \quad (30)$$

$$\textit{Mean return error} = \left(\sum_{j=1}^{51} 100|r_{i_j}^s - r_j^h|/r_j^h\right) \times \frac{1}{51} \quad (31)$$

### 4.2. Parameter tuning

To enhance the performance of the proposed algorithm, best parameter values are determined by using Taguchi method (Roy, 2001). In this method, the factors affecting response are divided into two categories, namely, signal and noise factors. Taguchi proposed a robust design approach to reduce the noise factors. In this study, Taguchi proposes signal to noise ($S/N$) ratios that measure the robustness. Purposely, larger $S/N$ ratios correspond to better the performance.

In this study, Taguchi method is implemented to the proposed algorithm by using FTSE100 data set available in OR Library. The levels of the algorithm parameters are presented in Table 1. The L27 orthogonal array is considered in experimental design. The performance measure of the algorithm ($R$) is presented in Eq. (32).

$$R = \left[\lambda\left[\sum_{i=1}^{N}\sum_{j=1}^{N} x_i x_j \sigma_{ij}\right] - (1-\lambda)\left[\sum_{i=1}^{N} x_i \mu_i\right]\right] \Big/ 51 \quad (32)$$

The main effects of $S/N$ ratios for the algorithm parameter levels are presented in Fig. 6. As one can infer from Fig. 6, best parameter values are $RCL_{size} = 0.4 \times N$, *minLocalSearch* = 0.2, *UB_minLocalSearch* = 25, and *randomFit* = 0.50.

### 4.3. Numerical results

In this section, numerical results obtained by the algorithm with tuned parameters are presented in comparison with the previously published results. For a fair comparison, *itrMAX_GRASP* value is determined as $25 \times N$ that yields approximate CPU times to the CPU times reported in related literature. As the previously published results are evaluated using different metrics and performance measures, in the current work, the results are compared to the corresponding studies as given separately in Tables 2–4. Fernández and Gómez (2007) utilize a neural network (NN) to solve the CCPO. In Table 2, the computational results of the proposed algorithm is compared to the results of Fernández and Gómez (2007) in terms of variance of return error, mean return error and mean percentage error (Eqs. (25)–(27)). As one can infer from the results, the proposed approach yield remarkably superior solutions for all instances.

On the other hand, the results are also compared with the results of Cura (2009) and Sadigh et al. (2012) in terms of mean Euclidian distance, variance of return error and mean return error (Eqs. (28)–(30)). Cura (2009) uses a particle swarm optimization algorithm (PSO) and Sadigh et al. (2012) develop a hybrid approach based on PSO and Hopfield neural network (PSO–HNN) to solve the CCPO. The results reveal that the proposed algorithm is competitive with the aforementioned algorithms (Table 3).

Numerical results demonstrate the effectiveness of the proposed approach in the current work. However, in the CCPO problem, the heuristic efficient frontiers are expected to scatter close to and in accordance with the standard efficient frontier. This feature of the found solutions is illustrated in Fig. 7, where dots represent those solutions of corresponding trade-off point defined by the level of $\lambda$. Therefore, a total of 51 trade-off points is observed on the plots in Fig. 7.

Moreover, the performance of the proposed algorithm is compared with the results reported by Woodside-Oriakhi et al. (2011) and Lwin and Qu (2013). Woodside-Oriakhi et al. (2011) used GA, TS and SA with subset optimization where Lwin and Qu (2013) develop a hybrid algorithm integrating population based

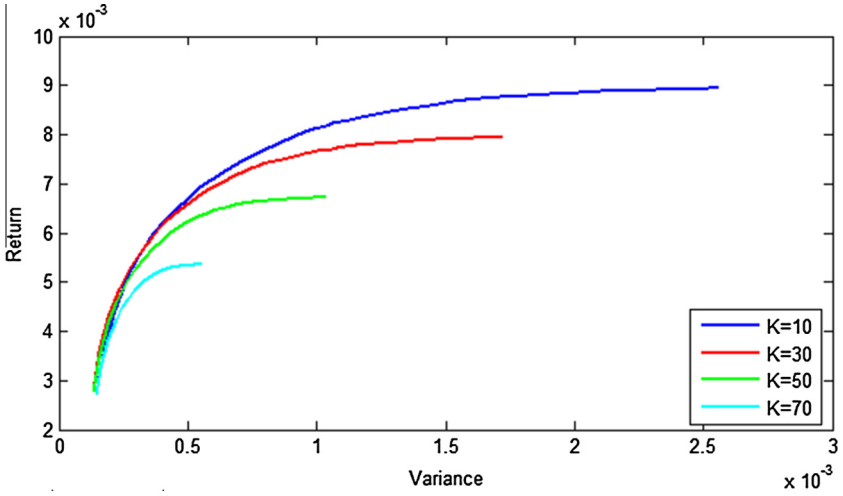



**Fig. 9.** Heuristic efficient frontiers for S&P 100 obtained for different cardinality values.

incremental learning and DE (PBILDE) to solve the CCPO problem. However, they handle the CCPO model in a different way from the objective weighting scheme presented in Eq. (8). They define the expected return of the portfolio as an equality constraint rather than an objective. Afterwards, they try to solve single objective variance minimization problem subject to different desired return levels. In other words, desired return levels are used to trace out the heuristic efficient frontier. The reader could find a detailed description of the approach in Woodside-Oriakhi et al. (2011). However, they relax the CCPO problem by reformulating the expected return constraint in which the expected return may vary within 10% of the desired return range. They use 50 equally spaced desired return levels between the expected return level associated with the minimum variance unconstrained portfolio and the return level associated with the maximum stock return. We used the same approach summarized here to compare mean (Eq. (27)), median, minimum and maximum percentage errors. According the results presented in Table 4, the proposed algorithm provides better solutions in majority of the benchmarking cases. Moreover, it can be asserted that the proposed algorithm yields robust solutions, as the mean and median percentage errors are found to be very close to each other generally.

Furthermore, heuristic efficient frontiers obtained in cases of different cardinality values ($K$) for Hang Seng and S&P data set are presented in Figs. 8 and 9. As one can see from the figures, heuristic efficient frontier diverges from standard efficient frontier and gets shorter as $K$ increases. In other words, both expected return and variance of portfolio decreases as the number of stocks in portfolio increases. As one focus on a fixed expected return level, the portfolios containing more stocks have relatively lower variance. This result is in parallel with diversification concept in portfolio management.

## 5. Conclusions

In this study, the CCPO is decomposed into two levels, namely, stock selection and proportion determination levels. In the stock selection level, a predetermined number of stocks are selected by using GRASP. Once the selection of the stocks by GRASP is completed, the problem turns out to be determining the proportions of the selected stocks. In the second level, an appropriate quadratic programming optimization solver is employed to find the proportions of the selected stocks. These two steps (levels) are iterated consecutively for a certain number of times and the final portfolio is obtained.

As a general point of view, the proposed solution methodology GRASP-QUAD hybridizes a meta-heuristic algorithm with an exact solution approach. The main advantage of the proposed approach is decomposition of the CCPO problem into two interrelated levels and thus, both levels become convenient to be solved faster and easily. Because GRASP selects a predetermined number of stocks, cardinality constraints are ensured. Moreover, quadratic solver handles the remaining constraints itself. Therefore, any further constraint handling procedures are not required.

According to the results, the proposed approach yields favorable solutions in comparison with the best known results published in the related literature. In regard to all of the findings presented throughout this paper, the proposed algorithm is found as an effective and promising approach for the CCPO problem.

## References

Anagnostopoulos, K. P., & Mamanis, G. (2009). Finding the efficient frontier for a mixed integer portfolio choice problem using a multiobjective algorithm. *iBusiness, 1*(2), 99–105. http://dx.doi.org/10.4236/ib.2009.12013.

Anagnostopoulos, K. P., & Mamanis, G. (2010). A portfolio optimization model with three objectives and discrete variables. *Computers & Operations Research, 37*(7), 1285–1297. http://dx.doi.org/10.1016/j.cor.2009.09.009.

Anagnostopoulos, K. P., & Mamanis, G. (2011). The mean–variance cardinality constrained portfolio optimization problem: An experimental evaluation of five multiobjective evolutionary algorithms. *Expert Systems with Applications*, 14208–14217. http://dx.doi.org/10.1016/j.eswa.2011.04.233.

Beasley, J. E. (1990). OR-Library <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/portinfo.html> Retrieved 2013.

Chang, T.-J., Meade, N., Beasley, J. E., & Sharaiha, Y. M. (2000). Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research, 27*, 1271–1302.

Cura, T. (2009). Particle swarm optimization approach to portfolio optimization. *Nonlinear Analysis: Real World Applications, 10*(4), 2396–2406. http://dx.doi.org/10.1016/j.nonrwa.2008.04.023.

Deng, G.-F., Lin, W.-T., & Lo, C.-C. (2012). Markowitz-based portfolio selection with cardinality constraints using improved particle swarm optimization. *Expert Systems with Applications, 39*(4), 4558–4566. http://dx.doi.org/10.1016/j.eswa.2011.09.129.

Feo, T., & Resende, M. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization, 6*(2), 109–133. http://dx.doi.org/10.1007/BF01096763.

Fernández, A., & Gómez, S. (2007). Portfolio selection using neural networks. *Computers & Operations Research, 34*(4), 1177–1191. http://dx.doi.org/10.1016/j.cor.2005.06.017.

Gupta, P., Mehlawat, M. K., & Mittal, G. (2011). Asset portfolio optimization using support vector machines and real-coded genetic algorithm. *Journal of Global Optimization, 53*(2), 297–315. http://dx.doi.org/10.1007/s10898-011-9692-3.

Lwin, K., & Qu, R. (2013). A hybrid algorithm for constrained portfolio selection problems. *Applied Intelligence, 39*(2), 251–266. http://dx.doi.org/10.1007/s10489-012-0411-7.

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance, 7*(1), 77–91.

Resende, M. C., & Ribeiro, C. (2010). Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In M. Gendreau & J.-Y. Potvin (Eds.). *Handbook of metaheuristics* (vol. 146, pp. 283–319). Berlin: Springer.

Roy, R. K. (2001). *Design of experiments using the Taguchi approach: 16 steps to product and process improvement*. John Wiley & Sons.

Sadigh, A. N., Mokhtari, H., Iranpoor, M., & Ghomi, S. M. T. F. (2012). Cardinality constrained portfolio optimization using a hybrid approach based on particle swarm optimization and Hopfield neural network. *Advanced Science Letters, 17*(1), 11–20.

Woodside-Oriakhi, M., Lucas, C., & Beasley, J. E. (2011). Heuristic algorithms for the cardinality constrained efficient frontier. *European Journal of Operational Research, 213*(3), 538–550. http://dx.doi.org/10.1016/j.ejor.2011.03.030.