2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Combinatorial Algorithm for The Cardinality Constrained Portfolio Optimization Problem

Tianxiang Cui*, Shi Cheng*†, and Ruibin Bai*
*Division of Computer Science, The University of Nottingham Ningbo, China
†International Doctoral Innovation Centre, The University of Nottingham Ningbo, China
{tianxiang.cui, shi.cheng, ruibin.bai}@nottingham.edu.cn

***Abstract*—Portfolio optimization is an important problem based on the modern portfolio theory (MPT) in the finance field. The idea is to maximize the portfolio expected return as well as minimizing portfolio risk at the same time. In this work, we propose a combinatorial algorithm for the portfolio optimization problem with the cardinality and bounding constraints. The proposed algorithm hybridizes a metaheuristic approach (particle swarm optimization, PSO) and a mathematical programming method where PSO is used to deal with the cardinality constraints and the math programming method is used to deal with the rest of the model. Computational results are given for the benchmark datasets from the OR-library and they indicate that it is a useful strategy for this problem. We also present the solutions obtained by the CPLEX mixed integer program solver for these instances and they can be used as the criteria for the comparison of algorithms for the same problem in the future.**

***Index Terms*—Cardinality Constrained Portfolio Optimization; Constrained problem; Particle swarm optimization.**

I. Introduction

One of the practical problems in fund management is how to allocate the limited capital to invest in a number of potential assets (investments) in order to achieve investors risk appetites and the return objectives. This often refers to the portfolio selection problem or the portfolio optimization problem. In the 1950s, Harry Markowitz proposed the foundations of modern portfolio theory (MPT) [1], [2]. In his work, the selection of the portfolio can be viewed as a mean-variance optimization problem with two criteria: to maximize the return of a portfolio and to minimize its risk. The portfolios meet both criterions are known as the efficient portfolios. Compared to an efficient portfolio, there should be no other portfolio with a higher expected return for a certain level of risk, or given the certain return level, there should be no other portfolio with a lower risk. For a particular universe of assets, the set of efficient portfolios forms the efficient frontier which represents the best trade-offs between the expected return and the risk.

The basic Markowitz mean-variance model needs to be expanded as it omits some real-world constraints. For example, it assumes that there exists a perfect market with no tax or transaction cost and short selling is not allowed. Also the assets should be infinitely divisible (i.e. they can be traded in any non-negative proportion). Normally, two common constraints in the real world problems need to be added: the cardinality constraint and the bounding constraints. The cardinality constraint specifies the total number of the held assets in a portfolio in order to reduce the tax and the transaction costs. The bounding constraints specify the lower and upper bound of the proportion of each held asset in a portfolio in order to avoid unrealistic holdings.

The closed form solution of the basic Markowitz mean-variance model can be efficiently found by common quadratic programming (QP). Some researchers try to apply the exact methods [3], [4] to the extended constrained model but they fail to obtain the optimal solutions in a reasonable computing time. The reason is that the introduction of the cardinality constraint changes the problem from a quadratic optimization model to a quadratic mixed-integer problem (QMIP). And as it has been proved in [5], it is an NP-hard problem.

Therefore, many researchers applied different heuristics and metaheuristic optimization techniques for the constrained model. Some classical methods have been adopted. For example Fernández *et al.* [6] applied the neural network method, Chang *et al.* [7] used simulated annealing (SA), tabu search (TS) and genetic algorithms (GA) and Woodside *et al.* [8] investigated the same three methods but used a revised neighborhood in order to get a better performance. Some hybrid methods also have been proposed. Di Gaspero *et al.* [9] combined the local search and quadratic programming together. Lwin *et al.* [10] integrated the population based incremental learning (PBIL) and differential evolution (DE). The experimental results of the hybrid approaches were competitive with the classical metaheuristics methods in terms of accuracy and speed.

In this work, we combine the strengths of the metaheuristic technique (namely particle swarm optimization, PSO) with the mathematical programming method and propose a combinatorial algorithm for the cardinality constrained portfolio optimization problem. PSO is applied for the assets selection (i.e. cardinality constraint) while a mathematical method is used to solve the rest of the model (i.e. bounding constraints, budget constraint). There are many kinds of variants of PSO algorithms for the same problem in the literature, the main difference here is that, other works apply PSO for the whole problem while we only use PSO to handle the cardinality constraint and the rest of the model can be solved in an exact manner. Therefore our method can guarantee the optimal allocation for a given assets combination.

The outline of the rest parts is as follows: in section II we give the statement of the problem and in section III we provide

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

a detailed description of our combinatorial algorithm. We give the datasets and our parameter settings in section IV and the computational results in section V. Section VI presents the results obtained by CPLEX for this problem and conclusions are given in section VII.

II. Problem Statement

A. The basic unconstrained mean-variance model

The basic Markowitz mean-variance model is formulated as the follows [1]:

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \sigma_{ij} \\ \text{subject to } & \sum_{i=1}^{N} w_i \mu_i = R^* \\ & \sum_{i=1}^{N} w_i = 1 \\ & 0 \le w_i \le 1, i = 1, \cdots, N \end{aligned}$$

where N is the number of assets available, μ_i is the expected return of the asset i $(i = 1, \ldots, N)$, σ_{ij} is the covariance between assets i and j $(i = 1, \ldots, N; j = 1, \ldots, N)$, R^* is the given expected return and w_i is the proportion $(0 \le w_i \le 1)$ of asset i $(i = 1, \ldots, N)$ held in the portfolio.

Based on the concept of Pareto optimality [11], a portfolio is called the efficient portfolio if for a given level of expected return, there is no other portfolio with a lower risk or for a given level of risk, there is no other portfolio with a higher expected return. The whole set of efficient portfolios forms the efficient frontier which represents the best possible mean returns for its risk levels. This efficient frontier is often referred to as the unconstrained efficient frontier (UEF) (see Figures 1,2,3,4).

B. The extended mean-variance model with the cardinality and bounding constraints

The basic Markowitz mean-variance model in the previous section omits some real world constraints and therefore it does not have too much use in practice. As the result, the basic model needs to be extended to address the real world problems. Usually, two types of trading constraints need to be added, which are the cardinality constraint and the bounding constraints. The cardinality constraint limits the maximum number of the assets within in a portfolio in order to reduce tax or transaction costs and the bounding constraints are the upper and lower bounds of the proportion of held assets within a portfolio.

Formally, the extended mean-variance model can be formulated as follows [7]:

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \sigma_{ij} \\ \text{subject to } & \sum_{i=1}^{N} w_i \mu_i = R^* \\ & \sum_{i=1}^{N} w_i = 1 && (1) \\ & \sum_{i=1}^{N} s_i = K && (2) \\ & \epsilon_i s_i \le w_i \le \delta_i s_i, \quad i = 1, \cdots, N && (3) \\ & s_i \in \{0, 1\}, \quad i = 1, \ldots, N && (4) \end{aligned}$$

where K is the desired number of the assets in the portfolio, s_i is a binary decision variable. If $s_i = 1$, asset i is chosen to be held. Otherwise, $s_i = 0$. If asset i is held, ϵ_i is the minimum proportion and δ_i is the maximum proportion. The proportion of the held asset i should lie in $[\epsilon_i, \delta_i]$ where $0 \le \epsilon_i \le \delta_i \le 1$.

This extended model is a quadratic mixed-integer problem (QMIP) which is NP-hard. Therefore, many researchers choose to use heuristic or metaheuristic in this area. For this work, we want to combine the strengths of both metaheuristic techniques and mathematical programming methods and present a hybrid approach which combines PSO and an exact method together to solve the extended constrained model.

III. The Proposed Combinatorial Algorithm

The most difficult part of the constrained model is the cardinality constraint (Equation (2)). It is mainly because the cardinality constraint is discrete and therefore the solution space is discontinuous. In the current literature, many researchers relax the cardinality constraint from an equality to an inequality (i.e. $\sum_{i=1}^{N} s_i \le K$). Apart from that, the rest of the model can be efficiently solved by using some commercial mathematical integer programming solvers. The idea of our approach is to combine the metaheuristic algorithm and mathematics method into a hybrid algorithm. The metaheuristic algorithm can be used to deal with the discrete constraint (cardinality) and mathematical method can be used to solve the rest of the model.

Usually, the quality of an individual solution in the metaheuristic algorithm is computed by using a fitness measure or a fitness function. Therefore, for each portfolio which satisfies the cardinality constraint, we can use a fitness function to calculate its quality. And then by iterations, we can get the portfolio with the better fitness value. For this work, we choose to use particle swarm optimization to handle the cardinality constraint and the Matlab math library to implement the corresponding fitness function.

A. Particle swarm optimization

Particle swarm optimization (PSO) algorithm, which is a nature-inspired searching techniques, was invented by Eber-

hart and Kennedy in 1995 [12], [13]. It is a population-based stochastic algorithm modeled on the social behaviors observed in flocking birds. Each particle, which represents a solution, flies through the search space with a velocity that is dynamically adjusted according to its own and its companion's historical behaviors. The particles tend to fly toward better areas of search space over the course of the search process [14].

The canonical PSO algorithm is simple in concept and easy in implementation [15], [16]. The basic equations are as the follows:

$$\mathbf{v}_i \leftarrow w\mathbf{v_i} + c_1\text{rand}()(\mathbf{x}_{pb} - \mathbf{x}_i) + c_2\text{Rand}()(\mathbf{x}_{lb} - \mathbf{x}_i) \quad (5)$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i \quad (6)$$

The meaning of notations are as follows:

- w denotes the inertia weight and is usually less than 1;
- c_1 and c_2 are two positive acceleration constants,
- rand() and Rand() are two independent functions to generate uniformly distributed random numbers in the range $[0, 1]$.
- $\mathbf{x}_i$ is the ith particle's position, which represents a solution to the problem.
- $\mathbf{v}_i$ is the ith particle's velocity,
- $\mathbf{x}_{pb}$ is personal best, which refers to the best position found by the ith particle, and
- $\mathbf{x}_{lb}$ is local best, which refers to the position found by the members in the ith particle's neighborhood that has the best fitness evaluation value so far.

The procedure of PSO algorithm is given as Algorithm 1.

Algorithm 1: The procedure of particle swarm optimization algorithm

```
Initialize velocity and position randomly for each particle in every dimension;
while not find the "good enough" set of assets or not reach the maximum number of iterations;
do
    Calculate each particle's fitness value;
    Compare fitness value between that of current position and that of the best position in history (personal best, termed as x_pb). For each particle, if the fitness value of current position is better than x_pb, then update x_pb to be the current position;
    Select the particle which has the best fitness value among current particle's neighborhood, this particle is called the neighborhood best (termed as x_lb).;
    for each particle do
        Update particle's velocity and position according to the Equation (5) and (6), respectively;
```

B. Problem Representation

In this paper, particle swarm optimization algorithm is utilized to solve a discrete problem with cardinality constraints. The search space (characterized by N in (1) and (2)) is different for different benchmark datasets (see Section IV-A). The aim of the search algorithm is to find the best k items from N instances. The details of problem representation are as follows:

- The fitness function f maps from a list of K integers to a real number: $\mathcal{Z}^K \rightarrow \mathcal{R}$ where K is the cardinality of the portfolio.
- A widely used representation is modelling portfolio selection problem as a N dimensional search problem [17]. A vector $\mathbf{z}$ represents the selection of portfolio, $z_i = 1$ means ith asset is selected and $z_i = 0$ otherwise. The dimension of algorithm's search space is increased with the increase of dataset's dimension. In our approach, the problem is modeled as a K dimensional search problem. A fixed length vector of $\mathbf{k} =< k_0, k_1, ..., k_K >$ is used to encode the selected K assets of the portfolio and $k_i \in \{1, 2, \ldots, N\}$. The selection of the ith item is dependent on other items, i.e., the search problem is a dimensional dependent problem.
- The swarm is a set of particles denoted by P. $\mathbf{x}_i$ of a particle $p_i \in P$ $(i = 1, \cdots, n)$ represents a potential solution. Each particle p_i consists of the two components:
  1) Position $\mathbf{x}_i = (x_1, x_2, \ldots, x_K)$, $\mathbf{x}_i \in \mathcal{Z}^K$ at step t. Each position is a list of K assets.
  2) Velocity $\mathbf{v}_i = (v_1, v_2, \ldots, v_K)$, $\mathbf{v}_i \in \mathcal{Z}^K$.
- The global best position is $\mathbf{x}_{\text{gb}}(t)$ such that $f(\mathbf{x}_{\text{gb}}(t)) \leq f(\mathbf{x}_i(t))$ for all $\mathbf{x}_i(t), p_i \in P$ at step t.

C. The fitness function

Once PSO chooses a set of assets which satisfies the cardinality constraint, we need a fitness function to determine the quality of the selected set (portfolio). We use Matlab to implement the fitness function as there are some useful built-in functions (they are used to handle the budget constraint (1) and bounding constraints (3)) in its toolbox and it is also more convenient to deal with the covariance matrix σ_{ij}. Once the fitness function is implemented, it can be built into a .DLL component which can be called by our PSO program procedure.

For the preprocessing part, we use Matlab to solve the basic mean-variance model in order to get the unconstrained efficient frontier (UEF). This UEF is the upper bound of the efficient frontier with the cardinality and bounding constraints being included.

Firstly, we create a new portfolio with the selected assets by PSO and set up the expected return and covariance matrix for that. Then we can set up the budget constraint (Equation (1)) and the bounding constraints (Equation (3)). After that, the efficient frontier of the new portfolio can be computed. This efficient frontier is called the sub-efficient frontier. The portfolios on the sub-efficient frontier satisfy all the constraints (Equation (1),(2),and (3)). Then we estimate m of optimal portfolios over the sub-efficient frontier.

Usually, determining the quality of a portfolio is done by using a percentage deviation method [7]. We apply the same

technique here. The percentage deviation error is measured by calculating the distance between the obtained portfolio and UEF, both horizontally and vertically.

Formally, let $(x_{\text{uef}}, y_{\text{uef}})$ be a discrete point on the UEF. The horizontal distance is calculated by taking the portfolio expected return as fixed ($y = y_{\text{uef}}$), linearly interpolating the point on the UEF to get the x value $x_{\text{interpolation}}$ and take the absolute value of the difference between x_{uef} and $x_{\text{interpolation}}$. Then the percentage deviation error in the x-direction is computed as $|x_{\text{uef}} - x_{\text{interpolation}}|/x_{\text{uef}} \times 100\%$. The percentage deviation error in the y-direction can be calculated in a similar way. The final percentage deviation error is the minimum of the percentage deviation error of the both directions.

We can choose m portfolios equally spaced on the UEF. These m portfolios represent m return levels. And for each chosen portfolio u_t on the UEF, we can find a portfolio u_s on the sub-efficient frontier with the smallest Euclidean distance to u_t. Then we compute the percentage deviation error of u_s and return the result as the fitness value.

For each portfolio u_s on the sub-efficient frontier, the selection of the assets are not necessarily the same. We call this fitness the dynamic set fitness. The procedure of the dynamic set fitness function is given in Algorithm 2.

Algorithm 2: The dynamic set fitness function

```
Input arguments: K asset numbers and a target portfolio u_t on the UEF;
Create a new portfolio with the selected K assets;
Set up the expected return and the covariance matrix of the new portfolio;
Set up the budget and bounding constraints;
Compute the sub-efficient frontier of the new portfolio;
Estimate m portfolios evenly distributed on the sub-efficient frontier;
Set u_min = u_1 where u_1 is the first portfolio on the sub-efficient frontier;
for each portfolio u_s on the sub-efficient frontier do
    Calculate the Euclidean distance between u_s and u_t;
    if The distance is less than the distance between u_min and u_t then
        Set u_min = u_s;
Compute the percentage deviation error of u_min and return the result;
```

We can also have a constant set fitness. The idea is to find the sub-efficient frontier which is "close to" the UEF. We calculate the sub-efficient frontier of the selected assets and then choose m portfolios equally spaced on the the sub-efficient frontier. We compute the percentage deviation error of these m portfolios and return the mean percentage deviation error as the fitness value. The procedure of the constant set fitness function is given as Algorithm 3.

In the dynamic fitness setting, for each different return level on the UEF we can get a portfolio P_d with the minimum

Algorithm 3: The constant set fitness function

```
Input arguments: K asset numbers;
Create a new portfolio with the selected K assets;
Set up the expected return and the covariance matrix of the new portfolio;
Set up the budget and bounding constraints;
Compute the sub-efficient frontier of the new portfolio;
Estimate m portfolios evenly distributed on the sub-efficient frontier;
for each portfolio u_s on the sub-efficient frontier do
    Compute the percentage deviation error of u_s;
Compute the mean percentage deviation error of m portfolios and return the result;
```

percentage deviation error. These obtained portfolios can also form a frontier. We choose m portfolios equally spaced on this frontier, label them from U_1^d to U_m^d and calculate the corresponding percentage deviation error from e_{d1} to e_{dm}. Similarly, in the constant fitness setting, we can compute a sub-efficient frontier. We also choose m portfolios equally spaced on this sub-efficient frontier, label them from U_1^c to U_m^c and calculate the corresponding percentage deviation error from e_1^c to e_m^c.

We then combine the results of both settings by taking the smaller percentage deviation error portfolio at each return level. Thus, if $e_i^d \leq e_i^c$, we say portfolio U_i^d dominates portfolio U_i^c and we take U_i^d; if $e_i^c < e_i^d$, we say portfolio U_i^c dominates portfolio U_i^d and we take $U_i^c (i = 1 \dots m)$.

IV. Data Set and Parameter Settings

A. The data set

A benchmark data set is available from the OR-library [18]. It contains the mean expected return and the correlation matrix of the following five different capital market indices:

- Hang Seng in Hong Kong, $N = 31$.
- DAX 100 in Germany, $N = 85$.
- FTSE 100 in UK, $N = 89$.
- S&P 100 in US, $N = 98$.
- Nikkei 225 in Japan, $N = 225$.

B. Parameters settings

- Standard PSO: We set $w = 0.72984$, $c_1 = c_2 = 1.496172$. These values are same as in the standard PSO [19], [20]. The population size of the particles is 48 and the number of iterations is 500 for each return level in every run. There are 5 runs in total for a single return level.
- Fitness function: We set $K = 10$, $\epsilon_i = 0.01$, $\delta_i = 1$ $(i = 1, \dots, N)$ and $m = 50$.

V. Computational Results

A. The unconstrained model

It is commonly to test the algorithm on the basic unconstrained model first in order to evaluate its performance [7],

[10]. For our case, if we set $K = N$ (N is the size of the dataset), $\epsilon_i = 0$ and $\delta_i = 1$, our algorithm can solve the basic unconstrained model in an exact manner (i.e. MPE = 0).

B. The constrained model

1) The dynamic fitness setting: The computational results of the dynamic fitness setting is shown in Table I. Here BPE denotes the best percentage deviation error, MedPE denotes the median percentage deviation error and MPE denotes the mean percentage deviation error.

TABLE I
COMPUTATIONAL RESULTS USING THE DYNAMIC FITNESS

Instance Index	N	BPE(%)	MedPE(%)	MPE(%)
Hang Seng	31	0.0148	0.0265	0.8725
DAX 100	85	0.7213	0.8467	2.1794
FTSE 100	89	0.2602	0.4369	1.3996
S&P 100	98	0.2969	0.3553	1.5352
Nikkei 225	225	0.2609	0.6651	0.7104
Average		0.3108	0.4661	1.3394

Figure 1 shows the comparative results of the portfolios obtained by our algorithm against the unconstrained efficient frontier (UEF) for five data sets in the dynamic fitness setting.

2) The constant fitness setting: The computational results of the constant fitness setting is shown in Table II. Figure 2 shows the comparative results of the sub-efficient frontier obtained by our algorithm against the unconstrained efficient frontier (UEF) for five data sets in the constant fitness setting.

TABLE II
COMPUTATIONAL RESULTS USING THE CONSTANT FITNESS

Instance Index	N	BPE(%)	MedPE(%)	MPE(%)
Hang Seng	31	0.1348	1.0412	0.9888
DAX 100	85	0.6692	1.1867	2.2455
FTSE 100	89	0.5541	0.9757	1.2137
S&P 100	98	0.3453	1.0773	1.2373
Nikkei 225	225	0.4289	0.7283	1.3966
Average		0.4265	1.0018	1.4163

3) The final results: We combine the results in both settings by taking the smaller percentage deviation error portfolio at each return level. We compare our results with those by Chang *et al.* [7] (as shown in Table III) and the best results so far in the literature [8] (as shown in Table IV). Figure 3 shows the comparison results of the final frontier obtained by our algorithm with the unconstrained efficient frontier (UEF).

4) The efficiency issue: The PSO algorithm was implemented in C#. All the tests were run on the same Intel(R) Core(TM) i5-3210M 2.50GHz processor with 4.00 GB RAM PC and Windows 8 system. The computational time of the whole program mainly depends on the fitness function implemented by Matlab. One interesting finding is, as Matlab has a powerful processing mechanism for the matrix, it takes almost the same amount of time to finish a single fitness calculation regardless of the size of the instances. For our case, it takes approximately 10 minutes to compute a single point for each different return level in the dynamic fitness setting and it takes approximately 60 minutes to compute the sub-efficient frontier in the constant fitness setting.

TABLE III
COMPARISON RESULTS OF OUR ALGORITHM WITH CHANG *et al.* [7] FOR THE CONSTRAINED MODEL

Index	Instance N		Chang -SA	Chang -TS	Chang -GA	Our method
Hang Seng	31	MedPE(%)	1.2082	1.1992	1.1819	**0.7546**
		MPE (%)	0.9892	0.9908	0.9457	**0.8250**
DAX 100	85	MedPE(%)	2.4675	2.5383	2.1262	**1.7032**
		MPE (%)	2.4299	3.0635	**1.9515**	2.1389
FTSE 100	89	MedPE(%)	0.7137	0.6361	**0.5938**	0.6548
		MPE (%)	1.1341	1.3908	0.8784	**0.8129**
S&P 100	98	MedPE(%)	1.1288	1.1487	1.1447	**1.0429**
		MPE (%)	2.6970	3.1678	1.7157	**1.5377**
Nikkei 225	225	MedPE(%)	0.6292	0.5914	0.6062	**0.5597**
		MPE (%)	0.6370	0.8981	0.6431	**0.6208**
Average		MedPE(%)	1.2294	1.2227	1.1306	**0.9430**
		MPE (%)	1.5774	1.9022	1.2269	**1.1870**

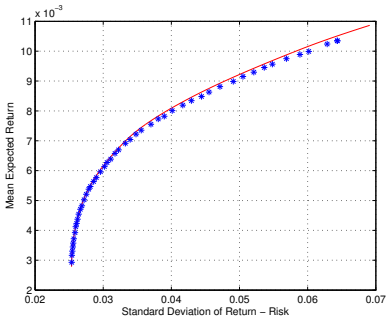
TABLE IV
COMPARISON RESULTS OF OUR ALGORITHM WITH WOODSIDE *et al.* [8] FOR THE CONSTRAINED MODEL

Index	Instance N		Woodside -SA	Woodside -TS	Woodside -GA	Our method
Hang Seng	31	MedPE(%)	0.5355	**0.3949**	0.5873	0.7546
		MPE (%)	1.0589	**0.8234**	0.8501	0.8250
DAX 100	85	MedPE(%)	0.8682	0.4298	**0.2400**	1.7032
		MPE (%)	1.0267	**0.7190**	0.7740	2.1389
FTSE 100	89	MedPE(%)	0.3944	0.2061	**0.0820**	0.6548
		MPE (%)	0.8952	0.3930	**0.1620**	0.8129
S&P 100	98	MedPE(%)	2.1064	1.0248	**0.1809**	1.0429
		MPE (%)	3.0952	1.0358	**0.2922**	1.5377
Nikkei 225	225	MedPE(%)	0.6877	0.6526	**0.3040**	0.5597
		MPE (%)	1.1193	0.7838	**0.3353**	0.6208
Average		MedPE(%)	0.9184	0.5416	**0.2788**	0.9430
		MPE (%)	1.4391	0.7510	**0.4827**	1.1870

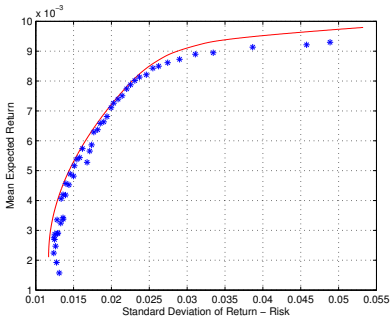
We also compared our results with the best ones so far in the literature by Woodside *et al.* [8]. Table IV shows Woodside-GA obtains better results for 4 out of 5 datasets compared to the ours. Our method outperforms Woodside-SA for 4 out of 5 instances and is competitive to Woodside-TS. The main difference between our method and Woodside *et al.* [7] is that our algorithm is a combinatorial algorithm which hybridizes both metaheuristics approach and mathematics method and it can guarantee the optimal allocation for a given assets combination. This may not be possible by using heuristics/metaheuristics techniques and it could lead to some promising future research directions.

VI. CPLEX SOLUTIONS

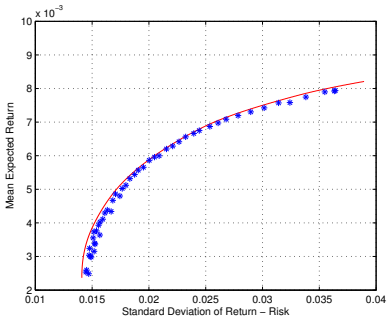
In order to test the effectiveness of our algorithm, we also use CPLEX (version 12.4) on the same datasets with the same



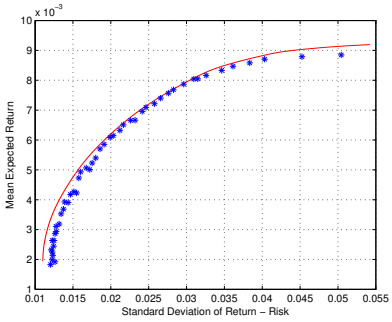
Hang Seng



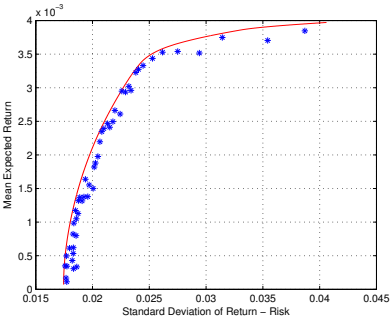
DAX 100



FTSE 100

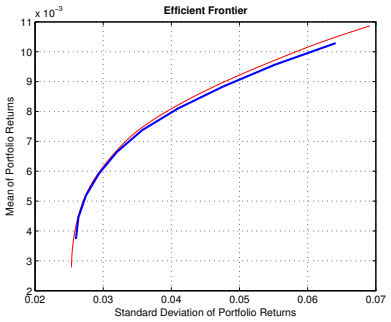


S&P 100

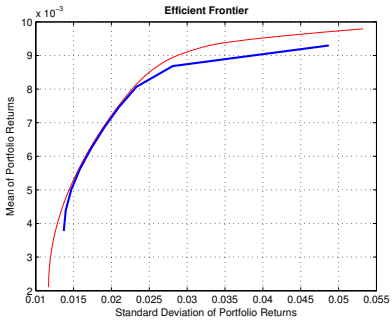


Nikkei 225

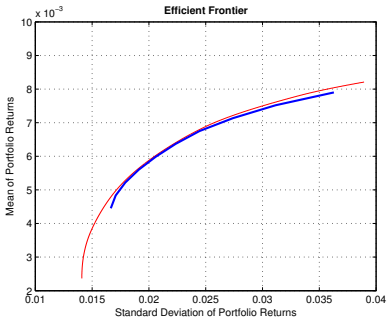
Fig. 1. Comparison of the portfolios obtained by our algorithm with the unconstrained efficient frontier (UEF) in the dynamic fitness setting. The upper curve is the UEF and the lower points are the portfolios found by our algorithm.



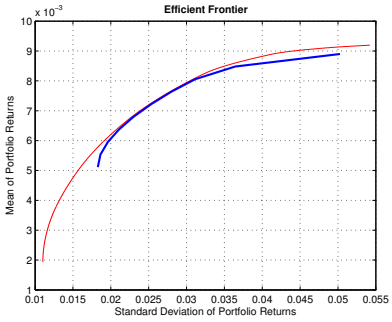
Hang Seng



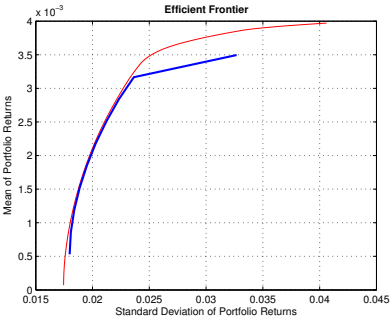
DAX 100



FTSE 100



S&P 100



Nikkei 225

Fig. 2. Comparison of the sub-efficient frontier obtained by our algorithm with the unconstrained efficient frontier (UEF) in the constant fitness setting. The upper curve is the UEF and the lower curve is the sub-efficient found by our algorithm.

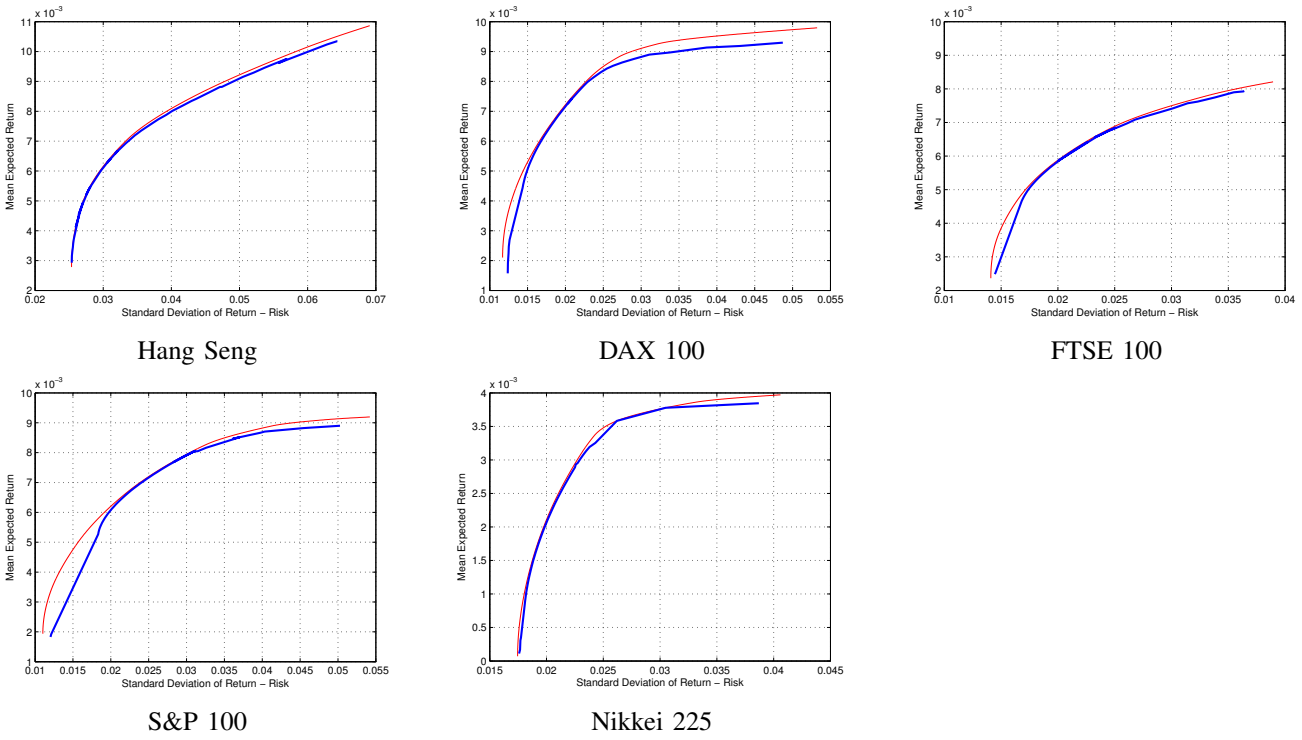


Fig. 3. Comparison results of the final frontier obtained by our algorithm with the unconstrained efficient frontier (UEF). The upper curve is the UEF and the lower curve is the final frontier found by our algorithm.

conditions ($K = 10$, $\epsilon_i = 0.01$, $\delta_i = 1$ $(i = 1, \ldots, N)$. We also choose 50 equally spaced return levels and for each return level we give CPLEX a time limit of 3600s on a same PC which was used to test our combinatorial algorithm. For some return levels, CPLEX obtains the solutions with a relaxed condition (feasible relaxed sum of infeasibilities), for some return levels, CPLEX can obtain the optimal or integer optimal solutions and for some return levels, CPLEX obtains the solutions with the time limit exceeded. The results are shown in Table V.

TABLE V
CPLEX RESULTS FOR THE CONSTRAINED MODEL

Instance Index	N	Number of optimal points obtained	MPE (optimal points)(%)	Average time per point (optimal)(s)
Hang Seng	31	47	0.6449	1.46
DAX 100	85	38	1.1851	3.23
FTSE 100	89	28	0.5022	2.37
S&P 100	98	27	0.6350	2.18
Nikkei 225	225	45	0.2449	1.67

Figure 4 shows the comparison of CPLEX results with the unconstrained efficient frontier (UEF). For the solutions with the time limit exceeded, it might be the case that CPLEX can not confirm the optimality of those points within the given time limit. For example the last point obtained with the time limit exceeded in DAX 100, it takes 10592s to confirm it is an integer optimal solution. Because of the limitation of the time, we did not test all of those points. Apart from the solutions with the time limit exceeded and a few infeasible solutions, we can see CPLEX can get optimal or integer solutions for most of the return levels. In the current literature, the reason of using MPE method to determine the quality of the algorithm is that there exists no optimal solutions so far, thus, if those CPLEX solutions' optimality can be confirmed, they can be used to determine the effectiveness of the algorithms for this problem in the future.

VII. Conclusion and Further Work

In this work, we propose a combinatorial algorithm which hybridizes a metaheuristic algorithm (PSO) and a mathematical programming method for the portfolio optimization problem with the cardinality and bounding constraints. PSO is used to deal with the cardinality constraints and the mathematical method is used to deal with the rest of the model. The main difference between our combinatorial algorithm and the heuristics/metaheuristics techniques in the literature is that our method can guarantee the optimal allocation for a given assets combination. The results are promising and indicate that it could be a useful strategy for the other finance and economics applications with similar models.

We also tested a mixed integer program solver CPLEX on the same datasets and the obtained results by CPLEX can be used as the criteria for comparing algorithms for this problem in the future.

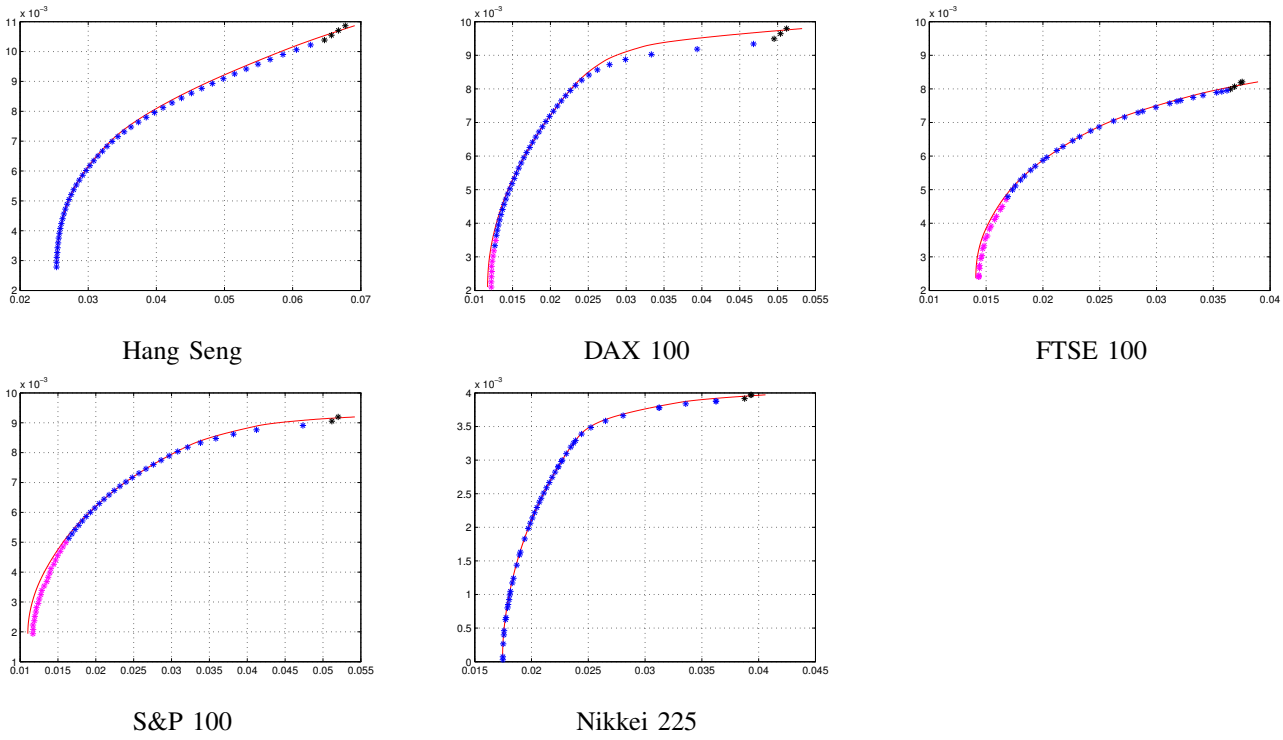


Fig. 4. Comparison of CPLEX results with the unconstrained efficient frontier (UEF). The red curve is the UEF. The black points are the ones with feasible relaxed sum of infeasibilities, the blue points are optimal or integer optimal points and the magenta points are the ones the time limit exceeded.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Grant No. 71001055), the Zhejiang Provincial Natural Science Foundation (Grant No. Y1100132) and Ningbo Science & Technology Bureau (Science and Technology Project No.2012B10055).

REFERENCES

[1] H. Markowitz, "Portfolio Selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, March 1952.

[2] H. M. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*, 2nd ed. Wiley, March 1991.

[3] D. Bienstock, "Computational study of a family of mixed-integer quadratic programming problems," *Mathematical programming*, vol. 74, pp. 121–140, 1996.

[4] D. Bertsimas and R. Shioda, "Algorithm for cardinality-constrained quadratic optimization," *Computational Optimization and Applications*, vol. 43, no. 1, pp. 1–22, May 2009.

[5] R. Moral-Escudero, R. Ruiz-Torrubiano, and A. Suárez, "Selection of optimal investment portfolios with cardinality constraints," in *Proceedings of the 2006 Congress on Evolutionary Computation (CEC2006)*, 2006, pp. 2382–2388.

[6] A. Fernández and S. Gómez, "Portfolio selection using neural networks," *Computers & Operations Research*, vol. 34, no. 4, pp. 1177–1191, April 2007.

[7] T. J. Chang, N. Meade, J. E. Beasley, and Y. M. Sharaiha, "Heuristics for cardinality constrained portfolio optimisation," *Computers & Operations Research*, vol. 27, no. 13, pp. 1271–1302, November 2000.

[8] M. Woodside-Oriakhi, C. Lucas, and J. E. Beasley, "Heuristic algorithms for the cardinality constrained efficient frontier," *European Journal of Operational Research*, vol. 213, no. 3, pp. 538–550, September 2011.

[9] L. Di Gaspero, G. Di Tollo, A. Roli, and A. Schaerf, "Hybrid metaheuristics for constrained portfolio selection problem," *Quantitative Finance*, vol. 11, no. 10, pp. 1473–1488, October 2011.

[10] K. Lwin and R. Qu, "A hybrid algorithm for constrained portfolio selection problems," *Applied Intelligence*, vol. 39, no. 2, pp. 251–266, September 2013.

[11] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.

[12] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks (ICNN)*, 1995, pp. 1942–1948.

[14] R. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001)*, 2001, pp. 81–86.

[15] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*. Morgan Kaufmann Publisher, 2001.

[16] R. Eberhart and Y. Shi, *Computational Intelligence: Concepts to Implementations*, 1st ed. Morgan Kaufmann Publisher, 2007.

[17] R. Ruiz-Torrubiano and A. Suárez, "Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 92–107, May 2010.

[18] J. E. Beasley, "OR-Library: distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069–1072, 1990.

[19] M. Clerc and J. Kennedy, "The particle swarm–explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, February 2002.

[20] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)*, April 2007, pp. 120–127.