This is a post-peer-review, pre-copyedit version of an article published in *Soft Computing*. The final authenticated version is available online at: https://doi.org/10.1007/s00500-019-04517-y

A Hybrid Combinatorial Approach to a Two-Stage Stochastic Portfolio Optimization Model With Uncertain Asset Prices

Tianxiang Cui*, Ruibin Bai*, Andrew J. Parkes†, Fang He§, Rong Qu†, Jingpeng Li‡
*Division of Computer Science, The University of Nottingham Ningbo, China
†School of Computer Science, The University of Nottingham, UK
‡Computing Science and Mathematics, University of Stirling, UK
§The Faculty of Science and Technology, University of Westminster, UK
Tianxiang.cui@nottingham.edu.cn

***Abstract*—Portfolio optimization is one of the most important problems in the finance field. The traditional mean-variance model has its drawbacks since it fails to take the market uncertainty into account. In this work, we investigate a two-stage stochastic portfolio optimization model with a comprehensive set of real world trading constraints in order to capture the market uncertainties in terms of future asset prices. Scenarios are generated to capture uncertain prices of assets. Stability tests are performed and the results confirm the effectiveness of the scenario generation method used for this work. We propose a hybrid combinatorial approach, which integrates a hybrid algorithm and a linear programming (LP) solver for the problem with a large number of scenarios, where the hybrid algorithm is used to search for the assets selection heuristically and the LP solver solves the corresponding sub-problems of weight allocation optimally. The hybrid algorithm is based on Population Based Incremental Learning (PBIL) while local search, hash search, elitist selection, partially guided mutation and learning inheritance are also adopted. Comparison results against other 3 algorithms are given. The results show that our hybrid combinatorial approach can solve the two-stage stochastic model effectively and efficiently. The effects of different parameter settings are also examined.**

***Index Terms*—Hybrid Algorithm; Combinatorial Approach; Stochastic Programming; Population-based Incremental Learning; Local Search; Learning Inheritance; Portfolio Optimization Problem.**

I. Introduction

With the advances in computing and the rise of big data, nowadays the investment decisions are made not only by the financial experts, but also based on sophisticated mathematical models and number crunching by mathematicians or computer scientists. The high yield of stock has made it a major investment over the past decades. One typical problem in stock market, portfolio optimization, can be described as allocating the limited capital over a number of potential assets in order to achieve investors risk appetites and the return objectives. The first portfolio optimization model was proposed by Markowitz in the 1950s [1], [2], where, the risk of the portfolio is measured as the variance of the asset return and therefore the problem can be viewed as a mean-variance optimization problem. The original problem is a quadratic programming problem, therefore it can be solved in an exact manner with a reasonable computational time.

However, the basic Markowitz mean-variance model has less practical utilities since it omits many constraints exist in real world trading. By imposing more real world constraints, for example cardinality (which specifies the total number of the held assets in a portfolio in order to reduce the tax and the transaction costs) and bounding (which specifies the lower and upper bound of the proportion of each held asset in a portfolio in order to avoid unrealistic holdings), the model can be transformed into an NP-complete problem [3], [4]. Our previous work [5] has proposed a combinatorial algorithm for the cardinality constrained portfolio optimization problem using the extended mean-variance model.

Although the real world constraints have later been introduced into the classic mean-variance model, there still remains another important market factor, the uncertainty, that complicates the investors making investment decisions. In the current work of mean-variance portfolio optimization problem [5]–[9], the mean expected return and the covariance between assets are assumed to be static, which is often unrealistic due to the economic turmoil and the market uncertainties in practice. It has been pointed out in [10], [11] that the investment decisions should be made based on the consideration of the market uncertainties. Usually, the random uncertainty factors are taken into account (i.e. the asset price, the currency exchange rate, the prepayments, the external cashflows, the inflation, the liabilities, etc.). There are also some other non-probabilistic uncertainty factors (i.e. the vagueness and the ambiguity, etc.) which are mainly modeled using fuzzy techniques [12]–[15]. In this work, we will focus on the random uncertainty of the market, more specifically, we consider the future asset prices to be uncertain.

Stochastic programming has been well studied for modeling optimization problems with uncertain factors since late 1950s [16]–[19]. It provides a stochastic view to replace the deterministic one in the sense that the uncertain factors are represented by the assumed probability distributions. It can model uncertainty and impose real world constraints in a flexible way

[20]. As it has been shown in [21], stochastic programming has been applied to many different areas successfully (finance, sports, scheduling [22], telecommunications, energy, production control and capacity planning, etc.). For this work, we propose to use stochastic programming to model uncertain future asset prices.

Another drawback of the mean-variance model is how it characterizes the risk. In the classical Markowitz portfolio optimization model, the risk is measured as the variance of the asset returns. Because such characterization of the risk is a measure of the dispersion of the values of the variable around its expected value, therefore it cannot define the direction of volatility in the sense that it penalizes the portfolio profits and the portfolio losses at the same time. Practically, people may only want to minimize the possibility of the portfolio losses. Alternatively, another risk measure, namely Value at Risk (VaR) [23], [24], is proposed to calculate the downside risk. VaR calculates the maximum possible loss with a specified confidence level and it is written into the industry regulation [25], [26]. However, VaR is inadequate for market risk evaluation since it does not satisfy the sub-additivity and the convexity and generally it is not a coherent risk measure [27]. Also, VaR does not take the distribution of the loss exceeding the threshold into account and it would become unstable if there is a sharp and heavy tail loss distribution. Furthermore, VaR is difficult to optimize using scenarios [28].

In order to eliminate the drawbacks of VaR, Rockafella and Uryasev [29] proposed Conditional Value at Risk (CVaR) which calculates the expected loss for the worst case scenarios. As it has been showed in [27], CVaR is a sub-additive and convex risk measure, therefore it can be optimized using stochastic programming.

Stochastic programming has been widely applied in financial optimization problems. Models for the management of fixed income securities [30]–[32] and models for asset/liability management [33]–[37] have been well studied in recent decades. A more comprehensive review can be found in [38]. A wide range of approaches based on stochastic programming for portfolio management have been developed [36], [37], [39]–[44].

In the portfolio optimization problem domain, Gaivoronski *et al.* [43] investigated different approaches to portfolio selection based on different risk characterizations. They proposed an algorithm to determine whether to rebalance a given portfolio based on transaction costs and new market condition information. Greco and Matarazzo [45] proposed an approach for portfolio selection in a non-Markowitz way. The uncertainties are modeled in terms of a series of meaningful quantiles of probabilistic distributions. They proposed an Interactive Multiobjective Optimization (IMO) method based on dominance-based Rough Set Approach (DRSA) to solve the model in two phases. Chen and Wang [46] introduced a hybrid stock trading system based on Genetic Network Programming and mean-CVaR model (GNP-CVaR). The proposed model combines the advantages of statistical models and artificial intelligence in the sense that CVaR measures the market risk and distributes the weights of capital to each asset in the portfolio and GNP decides the trading strategies. Stochastic programming models have also been widely used in the portfolio optimization literature. For example Topaloglou *et al.* [36] proposed a multi-stage stochastic programming model for international portfolio management in a dynamic setting. The uncertainties are modeled in terms of the asset prices and exchange rates. Yu *et al.* [47] proposed a dynamic stochastic programming model for bond portfolio management. They model the uncertainty in terms of the interest rates. Stoyan and Kwon [37] considered a stochastic-goal mixed-integer programming model for the integrated stock and bond portfolio problem. The uncertainties are modeled in terms of the asset prices and the real world trading constraints are imposed. The model was solved by a decomposition based algorithm. He and Qu [48] proposed a two stage portfolio selection problem with a comprehensive set of real world trading constraints. The uncertainties are modeled in terms of the asset prices. A hybrid algorithm integrating local search and a default Branch-and-Bound method was proposed to solve the problem.

One common method used in the literature to deal with stochastic portfolio optimization model is decomposition. Benders decomposition [44], scenario decomposition [49], time decomposition [42] and other novel decomposition methods [37] are proposed. The problem is simplified when it is decomposed into different parts.

In our previous work [50], we improved the stochastic portfolio optimization model in the literature [36], [48] and proposed a hybrid algorithm for the two-stage stochastic portfolio optimization problem with a comprehensive set of real world trading constraints. A genetic algorithm (GA) together with a commercial LP solver was used where GA is used to search for the assets selection heuristically and the LP solver can solve the corresponding sub-problems optimally. The proposed hybrid genetic algorithm can solve the problem to a good degree of accuracy, however, the full mechanisms of a standard GA is too heavy for the two-stage stochastic model, especially when a large number of scenarios is used. In order to solve the two-stage stochastic model more efficiently, in this work, we propose a light weight approach based on Population Based Incremental Learning (PBIL). It intends to solve the model with a larger number of scenarios. Local search, hash search, elitist selection and partially guided mutation are also adopted in order to enhance the evolution.

The outline of the rest part is as follows: section II introduces the background information. Section III gives the statement of the problem as well as the corresponding notations. The detail description of our hybrid combinatorial approach is given in section IV. The datasets are described in section V. In Section VI, we introduce our scenario generation method as well as the stability test results. Section VII gives the parameter settings and examines the effects of different learning rates. Experimental results are presented in section VIII. The final conclusion and possible future direction are given in section IX.

II. Preliminaries

A. Stochastic programming

In real-world situation, many variables are not deterministic due to the uncertain parameters involved (future events, human errors, etc). Generally, there are two approaches to deal with the uncertainties:

- *Robust Optimization*: when the uncertain variables are given within some certain boundaries, robust optimization is applied for such problems. The idea is to find a solution which is feasible for all the data and optimal for the worst case scenario.
- *Stochastic Programming*: when the probability distribution of the uncertain variables are known or can be estimated, stochastic programming is applied for such problems. The idea is to find a policy which is feasible for all (or at least almost all) possible data instances and maximizes/minimizes the expectation of the objective function with the decision and random variables involved. The decision-maker can gather some useful information by solving such models either analytically or numerically.

For this work, we use stochastic programming to deal with the uncertain future asset prices. The comprehensive concepts of stochastic programming can be found in [16], [17].

B. Two-Stage Stochastic Programming Problem With Recourse

For this work, we consider a widely applied class of stochastic programming problem, namely the recourse problem. It seeks a policy that can take the actions after some realisation of the uncertain variables as well as make the recourse decisions based on the temporarily available information.

The simplest case of the recourse problem have two stages:

- *first stage*: A decision needs to be made.
- *second stage*: The values of the uncertain variables are revealed and further decisions are allowed to make in order to avoid the constraints of the problem becoming infeasible. Usually a decision in the second stage will depend on a particular realisation of the uncertain variables.

Formally, the two-stage stochastic programming problem with recourse can be described as the follows [38]:

$$\begin{aligned} \min \quad & f(x) + E[\mathcal{Q}(x,\xi)] \\ \textit{s.t.} \quad & \\ & Ax = b \\ & x \in \mathbb{R}^n \end{aligned}$$

where ξ represents the uncertain data, x is the first-stage decision variable vector which should be decided before the uncertain variables are revealed and $\mathcal{Q}(x,\xi)$ is the optimal value for the following nonlinear program:

$$\begin{aligned} \min \quad & q(u,\xi) \\ \textit{s.t.} \quad & \\ & W(\xi)u = h(\xi) - T(\xi)x \\ & u \in \mathbb{R}^m \end{aligned}$$

where u is the vector of the second-stage decision variables which depends on the realization of the first-stage uncertain variables. $q(u,\xi)$ represents the second-stage cost function. $W(\xi)$, $h(\xi)$ and $T(\xi)$ are model parameters with reasonable dimensions. As these parameters are the functions of the uncertain data ξ, therefore they are also random. W is the recourse matrix and h is the second-stage resource vector. T is the technology matrix which contains the technology coefficients, therefore it can convert the first-stage decision variable vector x into resources for the second-stage problem.

Therefore the general two-stage stochastic programming problem with recourse can be rewritten as follows:

$$\begin{aligned} \min \quad & f(x) + E[\min\{q(u,\xi) | W(\xi)u + T(\xi)x = h(\xi)\}] \\ \textit{s.t.} \quad & \\ & Ax = b \\ & x \in \mathbb{R}^n \\ & u \in \mathbb{R}^m \end{aligned}$$

In this formulation, a "here and now" decision x is made before the uncertain data ξ is realized. At the second stage, after the value of the uncertain data ξ is revealed, we can modify our behavior by solving the corresponding optimization problem.

The recourse problem is not restricted to the two-stage formulation and it is possible to extend the problem into a multistage model.

C. Scenario Tree

There are two common methods which can be used to deal with the multistage stochastic programming problems, namely decision rule approximation and scenario tree approximation. For this work, we will focus on the scenario tree approximation tree method.

A scenario is defined as the possible realisation of the uncertain data ξ in each stage $t \in T$. An example of a scenario tree is showed in Figure 1. The nodes in the scenario tree represent a possible realisation of the uncertain data ξ^T. Each node is denoted by $n = (s,t)$ where s is a scenario and t is the level of the node in the tree and the decisions will be made at each node. The parent of the node n is represented by $a_{t-1}(n)$. The branching probability of the node n is denoted by p_n which is a conditional probability on its parent node $a_{t-1}(n)$. The path to the node n is a partial scenario with the probability $Pr_n = \prod p_n$ along the path and the sum of Pr_n is up to 1 across each level of the scenario tree.

In order to apply the scenario tree approximation method for the stochastic programming problem with recourse, the uncertain data ξ needs to be discretized and all possible realisations of ξ can be represented by a discrete set of scenarios. Thus, scenario generation methods are required. There are several scenario generation methods in the literature, for this work, we applied a shape based method [51].

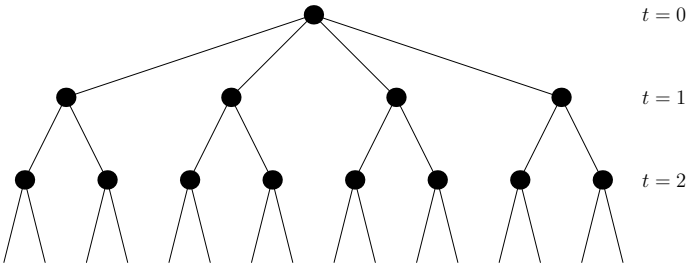


Fig. 1. An example of a scenario tree. At stage $t = 0$ there is one scenario, at stage $t = 1$ there are 4 scenarios and at stage $t = 2$ there are eight scenarios.

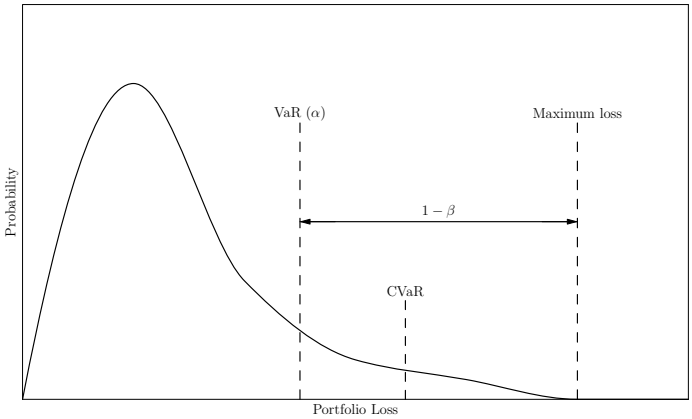


Fig. 2. Value at Risk (VaR) and Conditional Value at Risk (CVaR)

D. Percentile Risk Function

1) Value at Risk (VaR): In the real-world situation, portfolio managers may only need to reduce the possibility of the high loss. Value at Risk (VaR) [23], [24] gives the maximum possible loss α with a specified confidence level β. That is, by the end of the investing period, the probability of the loss exceeding the threshold α is $1 - \beta$ (see Figure 2).

Formally, let $f(x, \xi)$ be the loss function where $x \in \mathbb{Z}^+$ is the decision vector and $\xi \in \mathbb{R}$ is the uncertain (random) vector. The density of the probability distribution of ξ is denoted by $p(\xi)$. The probability of the loss function $f(x, \xi)$ not exceeding a threshold α is given by:

$$\Psi(x, \alpha) = \int_{f(x,\xi) \leq \alpha} p(\xi) d\xi$$

The β-VaR for the loss random variable associated with x and the specified probability β in $(0, 1)$ is denoted by $\alpha_\beta(x)$ and formally we have the following:

$$\alpha_\beta(x) = \min\{\alpha \in \mathbb{R} : \Psi(x, \alpha) \geq \beta\}$$

However, VaR is inadequate for market risk evaluation. As it has been pointed out in [27], VaR does not satisfy the sub-additivity and the convexity and generally it is not a coherent risk measure (VaR is only coherent for standard deviation of normal distributions). Also VaR is difficult to optimize using scenarios [28]. Furthermore, VaR does not take the distribution of the loss exceeding the threshold into account and it would become unstable if there is a sharp and heavy tail in the loss distribution.

2) Conditional Value at Risk (CVaR): Conditional Value at Risk (CVaR) (also called Mean Excess Loss, Mean Expected Shortfall, Tail VaR) is proposed in [29] in order to eliminate the drawbacks of VaR. CVaR is a more consistent risk measure because of its sub-additivity and the convexity [27] and it is proven to be a coherent risk measure [52].

CVaR calculates the average value of the loss which exceeds the VaR value (see Figure 2). Formally, CVaR is defined as the follows [29]:

$$\phi_\beta(x) = (1 - \beta)^{-1} \int_{f(x,\xi) \geq \alpha_\beta(x)} f(x, \xi) p(\xi) d\xi$$

The function above is a little bit difficult to handle because the VaR value $\alpha_\beta(x)$ is involved in it. Alternatively, we can have the analytical representation to replace VaR. A simpler function can be used instead of CVaR:

$$F_\beta(x, \alpha) = \alpha + (1 - \beta)^{-1} \int_{f(x,\xi) \leq \alpha} (f(x, \xi) - \alpha) p(\xi) d\xi$$

It has been proved in [29] that $F_\beta(x, \alpha)$ is a convex function with respect to α and the minimum point of $F_\beta(x, \alpha)$ is VaR with respect to α. The CVaR value can be obtained by minimizing $F_\beta(x, \alpha)$ with respect to α.

3) Minimizing CVaR: From the definitions of VaR and CVaR we can see that given a specified probability level β, β-CVaR should always be greater or equal to β-VaR. In fact, we can optimize CVaR and obtain VaR simultaneously by minimizing the function $F_\beta(x, \alpha)$ [53]. Suppose we have the solution of the minimization of $F_\beta(x, \alpha)$, (x^*, α^*), then the optimal CVaR value equals to $F_\beta(x^*, \alpha^*)$ and the corresponding VaR value equals to α^*.

We can minimize the function $F_\beta(x, \alpha)$ by introducing an auxiliary function $Z(\xi)$ such that $Z(\xi) \geq f(x, \xi) - \alpha$ and $Z(\xi) \geq 0$. Formally we have the following:

$$\begin{aligned} \min \quad & \alpha + (1 - \beta)^{-1} E(Z(\xi)) \\ \textit{s.t.} \quad & \\ & Z(\xi) \geq f(x, \xi) - \alpha \\ & Z(\xi) \geq 0 \\ & \alpha \in \mathbb{R} \end{aligned}$$

Now let us consider the portfolio optimization problem. Here the uncertain data ξ can be referred to the future asset prices. Normally the analytical representation of density function $p(\xi)$ is not available but instead the scenarios can be generated from the historical observations of each asset price. The scenario generation can use the property matching method [54], [55] or even simply Monte Carlo simulations. Suppose we have generated N scenarios from the density $p(\xi)$, y_n where $n = 1, \ldots, N$. Function $F_\beta(x, \alpha)$ can be therefore calculated as the follows:

$$F_\beta(x,\alpha) = \alpha + (1-\beta)^{-1}\sum_{n=1}^{N} p_n (f(x,y_i)-\alpha)^+$$

where $f(x,y_n)$ is the portfolio loss function in scenario n and it is defined as the negative of the total portfolio return. p_n is the probability of scenario n and $(f(x,y_i)-\alpha)^+ = \max(0,(f(x,y_i)-\alpha))$. By introducing the auxiliary function $Z(\xi)$ and we can have the auxiliary variable z_n where $z_n \geq f(x,y_n)-\alpha, z_n \geq 0, n = 1,\ldots,N$. Therefore the minimization of the function $F_\beta(x,\alpha)$ can be reduced to the simplified form:

$$\min \quad \alpha + (1-\beta)^{-1}\sum_{n=1}^{N} p_n z_n$$

s.t.

$$z_n \geq f(x,y_n) - \alpha \quad n = 1,\ldots,N$$
$$z_n \geq 0 \quad n = 1,\ldots,N$$
$$\alpha \in \mathbb{R}$$
$$x \in \mathbb{R}^n$$

It has been showed in [29], [56], [57] that such formulation can provide the numerically stable technique to the problem with large number of scenarios.

III. Model Statement

A. Notations

The notations we used in this work are given in Table I.

B. Two-stage stochastic portfolio optimization model with recourse

The model we used for this work is the same with our previous work [50]. Inspired by [36], the original form of the model was proposed in [48]. Although [48] is formulated as a two stage model, it did not include a possibility that the costs and values change after the recourse decision is enacted. Hence the recourse it that model could have no monetary effect, and so would obtain the same decisions as a simpler single stage formulation. A contribution of our work is to extend the model so that values can change after the recourse, and non- trivial recourse decisions can improve the portfolio performance. The proposed model is divided into two stages.

$$\min \left(\alpha + (1-\beta)^{-1}\sum_{j\in N_r} p_j z_j \right) \tag{1}$$

s.t.

First Stage - Portfolio Selection:

$$w_i = w_i^0 + b_i - s_i, \quad \forall i \in A \tag{2}$$
$$h + \sum_{i\in A}(s_i P_i^0) - \sum_{i\in A}(\eta_s g_i + s_i \rho_s P_i^0) = \sum_{i\in A}(b_i P_i^0) + \sum_{i\in A}(\eta_b f_i + b_i \rho_b P_i^0) \tag{3}$$
$$\sum_{i\in A} c_i = K \tag{4}$$
$$w_{\min} c_i \leq w_i \quad \forall i \in A \tag{5}$$
$$t_{\min} f_i \leq b_i \quad \forall i \in A \tag{6}$$
$$t_{\min} g_i \leq s_i \quad \forall i \in A \tag{7}$$
$$f_i + g_i \leq 1 \quad \forall i \in A \tag{8}$$
$$f_i M \geq b_i \quad \forall i \in A \tag{9}$$
$$g_i M \geq s_i \quad \forall i \in A \tag{10}$$
$$b_i M \geq f_i \quad \forall i \in A \tag{11}$$
$$s_i M \geq g_i \quad \forall i \in A \tag{12}$$
$$w_i,\ b_i,\ s_i \in \mathbb{R} \tag{13}$$
$$c_i,\ f_i,\ g_i \in \mathbb{B} \tag{14}$$

Second Stage - Recourse:

$$w_i^j = w_i + b_i^j - s_i^j \quad \forall i \in A,\ \forall j \in N_r \tag{15}$$
$$\sum_{i\in A}(s_i^j P_i^j) - \sum_{i\in A}(\eta_s g_i^j + s_i^j \rho_s P_i^j) = \sum_{i\in A}(b_i^j P_i^j) + \sum_{i\in A}(\eta_b f_i^j + b_i \rho_b P_i^j) \quad \forall j \in N_r \tag{16}$$
$$\sum_{i\in A} c_i^j = K \quad \forall j \in N_r \tag{17}$$
$$w_{\min} c_i^j \leq w_i^j \quad \forall i \in A,\ \forall j \in N_r \tag{18}$$
$$t_{\min} f_i^j \leq b_i^j \quad \forall i \in A,\ \forall j \in N_r \tag{19}$$
$$t_{\min} g_i^j \leq s_i^j \quad \forall i \in A,\ \forall j \in N_r \tag{20}$$
$$f_i^j + g_i^j \leq 1 \quad \forall i \in A,\ \forall j \in N_r \tag{21}$$
$$f_i^j M \geq b_i^j \quad \forall i \in A,\ \forall j \in N_r \tag{22}$$
$$g_i^j M \geq s_i^j \quad \forall i \in A,\ \forall j \in N_r \tag{23}$$
$$b_i^j M \geq f_i^j \quad \forall i \in A,\ \forall j \in N_r \tag{24}$$
$$s_i^j M \geq g_i^j \quad \forall i \in A,\ \forall j \in N_r \tag{25}$$
$$V^j = \sum_{\substack{i\in A \\ e\in N_e^j}} p_{(j,e)} P_i^{(j,e)} w_i^j \quad \forall j \in N_r \tag{26}$$
$$R^j = V^j - V^0 \quad \forall j \in N_r \tag{27}$$
$$z_j \geq -R^j - \alpha \quad \forall j \in N_r \tag{28}$$
$$z_j \geq 0 \quad \forall j \in N_r \tag{29}$$
$$\sum_{j\in N_r} p_j R^j \geq \mu \tag{30}$$
$$w_i^j,\ b_i^j,\ s_i^j \in \mathbb{R} \tag{31}$$
$$c_i^j,\ f_i^j,\ g_i^j \in \mathbb{B} \tag{32}$$
$$\alpha,\ z_j \in \mathbb{R} \tag{33}$$

TABLE I
NOTATIONS IN THE MODEL

Type of Data	Notation	Meaning
Set	A	The set of assets
Set	N_r	The set of recourse nodes. One node corresponds to one recourse portfolio
Set	N_e^j	The set of evaluate nodes on recourse node j where $j \in N_r$
User-specific parameter	μ	The target return
User-specific parameter	β	Quantile (percentile) for VaR and CVaR
User-specific parameter	M	The big constant
Deterministic input data	h	The initial cash to invest
Deterministic input data	w_i^0	The initial position of asset i (in number of units)
Deterministic input data	η_b	The fixed buying cost
Deterministic input data	η_s	The fixed selling cost
Deterministic input data	ρ_b	The variable buying cost
Deterministic input data	ρ_s	The variable selling cost
Deterministic input data	K	The number of asset held in the portfolio (cardinality)
Deterministic input data	w_{min}	The minimum holding position
Deterministic input data	t_{min}	The minimum trading size
Scenario dependent data	p_j	The probability of recourse node j in the second stage
Scenario dependent data	$p_{(j,e)}$	The probability of evaluate node e of recourse node j in the second stage
Scenario dependent data	P_i^0	The price of asset i in the first stage (per unit)
Scenario dependent data	P_i^j	The price of asset i on recourse node j in the second stage (per unit)
Scenario dependent data	$P_i^{(j,e)}$	The price of asset i on evaluate node e of recourse node j in the second stage (per unit)
Scenario dependent data	V^0	The initial portfolio wealth
Scenario dependent data	V^j	The final portfolio wealth on recourse node j
Auxiliary variable	z_j	Portfolio shortfall in excess of VaR at recourse node j
Auxiliary variable	α	The optimal VaR value
Decision variable	b_i	The number of units of asset i purchased in the first stage
Decision variable	s_i	The number of units of asset i sold in the first stage
Decision variable	w_i	The final position of asset i in the first stage
Decision variable	b_i^j	The number of units of asset i purchased on recourse node j in the second stage
Decision variable	s_i^j	The number of units of asset i sold on recourse node j in the second stage
Decision variable	w_i^j	The final position of asset i on recourse node j in the second stage
Decision variable	c_i	The binary holding decision variable in the first stage
Decision variable	f_i	The binary buying decision variable in the first stage
Decision variable	g_i	The binary selling decision variable in the first stage
Decision variable	c_i^j	The binary holding decision variable on recourse node j in the second stage
Decision variable	f_i^j	The binary buying decision variable on recourse node j in the second stage
Decision variable	g_i^j	The binary selling decision variable on recourse node j in the second stage

The objective function (1) calculates the β-percentile CVaR of the portfolio loss at the end of the second stage where α is the corresponding optimal VaR value. Eq. (2) is the first stage asset balance condition and Eq. (15) is the second stage asset balance condition. Equations (3), (16) are the cash balance conditions for the first and second stage respectively. We apply a fixed transaction cost and a linear variable transaction cost to both buying and selling an asset. The idea is that the cash inflows should equal to the cash outflows in both stages (i.e. no cash left). Equations (4), (17) are the cardinality constraints for the first and second stage respectively where K is the desired number of the assets held within a portfolio. Equations (5) and (18) put the restrictions on the minimum holding size of an asset in order to prevent very small asset positions for the first and second stages. Equations (6),(7), are the minimum trading conditions for the first stage and equations (19),(20), are the minimum trading conditions for the second stage. The idea is to prevent it from trading a very small proportion of an asset. Buying and selling the same asset at the same time is not allowed, this is given in equation (8) for the first stage and in equation (21) for the second stage. The big-M formulations are used in the model in order to bound the decision variables and the binary decision variables (constraints (9), (10), (11), (12) for the first stage and constraints (22), (23), (24), (25) for the second stage). The idea is, if the decision variables for buying/selling an asset is greater than 0, then the corresponding binary decision variables should equal to 1; if the decision variables for buying/selling an asset is 0, then the corresponding binary decision variables should be 0 and vice versa. Equations (26), (27) calculate the portfolio return on each recourse node by using a different set of evaluate scenarios in order to have a better reflection of changing price scenarios in the reality. Equations (28), (29) define the excess shortfall z_j of the recourse portfolio where $z_j = \max[0, -R^j - \alpha]$ for each recourse node. The minimum portfolio target return μ is given in equation (30). The decision variables $w_i, b_i, s_i, w_i^j, b_i^j, s_i^j$ specify the exact amount of the units for an asset to buy or sell and in a real-world situation, these decision variables should be integers. As they increase the computational difficulty significantly, we took the same method suggested in [48], [58] to relax these decision variables as continuous variables.

C. Computational complexity

The deterministic problems are generally in NP, however, the stochastic versions can be of even harder complexity classes; for example, it has been shown in [59] that linear two-stage stochastic programming problems are #P-hard. For multistage stochastic programming problems, it is generally computationally intractable even for the medium-accuracy solutions [60].

IV. The Proposed Hybrid Combinatorial Approach

Exact methods and metaheuristic approaches are two successful streams for solving combinatorial optimization problems. Over the last few years, many works have been developed on building hybrids of exact methods and metaheuristic approaches. In fact, many real-world problems can be practically solved much better using hybrid strategies since the advantages of both types of methods are simultaneously exploited. For this work, we integrate metaheuristic approaches with exact methods. The idea is, we divide the two-stage stochastic problem into two parts. The first part is to determine the asset combination while the second part is to calculate the optimal weights of the selected assets correspondingly. The first part is searched by the hybrid algorithm and the second part is solved by a LP solver. In our previous work [50], we proposed a hybrid genetic algorithm which can solve the problem to a good degree of accuracy. However, the full mechanisms of GA is too heavy (i.e. computationally expensive) for the two-stage stochastic model especially when a large number of scenarios are used. In this case, we propose a lighter approach which is based on Population Based Incremental Learning (PBIL). It intends to solve the two-stage stochastic model with a larger number of scenarios. Local search, hash search, elitist selection and partially guided mutation are also adopted in order to enhance the evolution.

A. Overview of PBIL

Population Based Incremental Learning (PBIL) was originally introduced by Baluja [61], [62]. It is one of the simplest form of Estimation of Distribution Algorithms (EDAs). It combines genetic algorithms and competitive learning for function optimization. It evolves the entire population rather than each single individual members. The idea is, a probability vector is used to represent the distribution of all individuals. After evaluating each individuals, the probability vector is updated by learning from the best and the worst solutions. Mutation is also performed on the probability vector in order to help preserve diversity. Then the new generation of population is created based on the updated probability vector. PBIL is closely related to GA, but it is simpler and more efficient since it does not require all the mechanisms of a standard GA.

B. Problem representation

In this work, PBIL-based hybrid algorithm is utilized to evolve best values for discrete variables in the stochastic model. The search space is different for different benchmark datasets (characterized by Q, see Section V). The objective is to find the best K items from Q possible assets for a given target return μ specified by the investor. The details of problem representation are as follows:

- One probability vector $\mathbf{v} = (v_0, v_1, ..., v_Q)$ of size Q represents the possibility of each asset to be chosen in the portfolio.
- One binary vector $\mathbf{t} = (t_0, t_1, ..., t_Q)$ of size Q is used to denote if asset is chosen in the portfolio.
- One vector $\mathbf{k} = (k_0, k_1, ..., k_K)$ of size K is used to represent the selected K assets of the portfolio where $k_i \in \{1, 2, \ldots, Q\}$ and $i = 1, \ldots, K$.
- The evaluation of vector $\mathbf{k}$ is done by calculating a fitness function F which is implemented using a standard LP solver. It maps from a list of K integers and a target return μ to a real number: $F(\mathcal{Z}^K, \mu) \rightarrow \mathcal{R}$.
- The probability vector is updated by learning from the best and the worst solutions obtained from the population at the end of each generation.
- Elitist selection is used in our PBIL-based hybrid algorithm, i.e., we keep the best solution in each generation.
- The global best solution $\mathbf{x}_{\text{gb}}$ is recorded such that $F(\mathbf{x}_{\text{gb}}, \mu) \leq F(\mathbf{x}_i, \mu)$ for all $\mathbf{x}_i$ at the given return level μ.

The procedure of PBIL-based hybrid algorithm used in this work is given as Algorithm 1 and the parameters are given in section VII.

Algorithm 1: PBIL-based hybrid algorithm for searching the set of assets

```
for i = 1 to Q do
    v_i = 0.5;
while stopping criteria are not met;
do
    Generating individuals: Create a population of individuals (see IV-D);
    for each individual generated do
        Hash search: Search the infeasible solution hash table and bad solution hash table, if it is very similar to the entries of the hash table, re-generating (see IV-E);
    Evaluation: Evaluate each individual's fitness by using CPLEX LP solver (see IV-F);
    Local Search: Perform the local search for the top 20% individuals (see IV-G);
    Archive: Keep the record of the current best solution, the current worst solution and the infeasible solution (see IV-H);
    Update: Update v by learning from the current best and the current worst solution (see IV-I);
    Mutation: Mutate v by using partially guided mutation (see IV-J);
    Elitism: Select the best individual from the current generation and insert it into the next new generation;
```

C. The reduced sub-problem

In this work, the sub-problems are generated by dropping all the non-selected assets. Each individual is fixed in both the first and second stage (i.e. $c_i = c_i^j = 1$ if hybrid algorithm picks asset i). The recourse are limited to asset rebalancing, but not swapping the assets and therefore we can call such sub-problem the reduced sub-problem. As the transaction costs and

the minimum holding constraint are considered in the model, the cost of buying an entirely new asset is probably significantly higher than just adjusting the holding of an existing one. We use CPLEX to solve a full problem on a small instance using a small number of scenarios with different transaction costs and minimum holding constraint. For each different set of transaction costs and minimum holding constraint, we count the number of occurrences that the selection of assets in the second-stage is not the same as in the first-stage (i.e. $c_i^j \neq c_i$). The results are shown in Table II. We can see that when the transaction costs are bigger than 0.4% and the minimum holding constraint is bigger than 0.8%, the assets in the second-stage remain unchanged. For this work, we use the parameter settings $\rho_b, \rho_s = 0.5\%$ $w_{min} = 1.0\%$ (see section VII) therefore adding or dropping the rebalance-only condition should not make a significant difference. We do not claim this is an optimal approximation. The reason of using the reduced sub-problem is to reduce the computation time by the LP solver.

TABLE II
THE NUMBER OF OCCURRENCES THAT $c_i^j \neq c_i$ FOR THE SOLUTION OF THE FULL PROBLEM ON A SMALL INSTANCE USING 100 POSSIBILITIES OF SCENARIOS WITH DIFFERENT MODEL PARAMETERS

Model parameters	Q	N_r	N_e^j	# of times that $c_i^j \neq c_i$
$\rho_b, \rho_s = 0.5\%$ $w_{min} = 1.0\%$	31	20	5	0
$\rho_b, \rho_s = 0.4\%$ $w_{min} = 0.8\%$	31	20	5	0
$\rho_b, \rho_s = 0.3\%$ $w_{min} = 0.6\%$	31	20	5	7
$\rho_b, \rho_s = 0.2\%$ $w_{min} = 0.4\%$	31	20	5	16
$\rho_b, \rho_s = 0.1\%$ $w_{min} = 0.2\%$	31	20	5	32

D. Generating individuals

The probability vector $\mathbf{v}$ is used to determine whether asset i is chosen in the portfolio. Initially v_i is set to 0.5 where $i = 1, \ldots, Q$ so that every asset can have an equal chance to be chosen. The binary vector $\mathbf{t}$ is created according to $\mathbf{v}$, if asset i is selected, $t_i = 1$ and if asset i is not selected, $t_i = 0$. Then vector $\mathbf{k}$ which is used to represent the chosen asset in the portfolio is generated according to vector $\mathbf{t}$. The idea is to chose exact K number of assets to form the portfolio in order to satisfy the cardinality constraint. Suppose there are K' assets selected in $\mathbf{t}$, if $K' \geq K$, we randomly choose K among K' assets and insert them into $\mathbf{k}$. If $K' < K$, we first insert K' assets into $\mathbf{k}$, then we randomly choose another $K - K'$ assets which are different from the existing K' assets and insert them into $\mathbf{k}$. The procedure is given as Algorithm 2. For each generation, a population of such individuals is created.

Algorithm 2: Generating individuals

```
for i = 1 to Q do
    if random(0,1) < v_i then
        t_i = 1;
    else
        t_i = 0;
K' = 0;
for i = 1 to Q do
    if t_i == 1 then
        K' = K' + 1;
if K' ≥ K then
    randomly choose K among K' assets and insert them into k;
if K' < K then
    insert K' assets into k;
    randomly choose another K − K' assets which are different from the existing K' assets and insert them into k;
```

E. Hash search

By using our two-stage stochastic model, for any given asset combination, it does not necessarily always lead to a feasible solution. We maintain a hash table to keep all the infeasible solutions explored. We also have a hash table to keep all the worst feasible solutions of each generation (see IV-H). Each time when a new individual is generated, we check its similarity with the existing entries in the hash tables. If it is very similar to the existing entries, we discard it and re-generate the individual. As it has been pointed out in [63], good solutions tend to have similar structures and bad solutions also tend to have the similar structures. There are two advantages of performing the hash search. Firstly the computational cost of the hash table lookup is amortized $O(1)$ ($O(1)$ on average, $O(n)$ for the worst case), which is cheaper than calling the LP solver, therefore it improves the efficiency; secondly it can explore different areas of the solution space by avoiding the unnecessary search, and it may explain why, in Figure 3, PBIL with the hash search tends to obtain better global solutions. The details of the hash search are given as Algorithm 3 and 4.

Algorithm 3: SimilarityCheck(String Str1, String Str2)

```
count = 0;
for each character char1 in Str1 do
    for each character char2 in Str2 do
        if char1 == char2 then
            count = count + 1 ;
return count ;
```

F. Evaluation

The fitness of the individual generated is evaluated by solving the corresponding sub-problem using an LP solver in order to get the weight allocation of the selected assets. We can control the numerical properties of the solutions to the sub-problems by setting up different Markowitz threshold

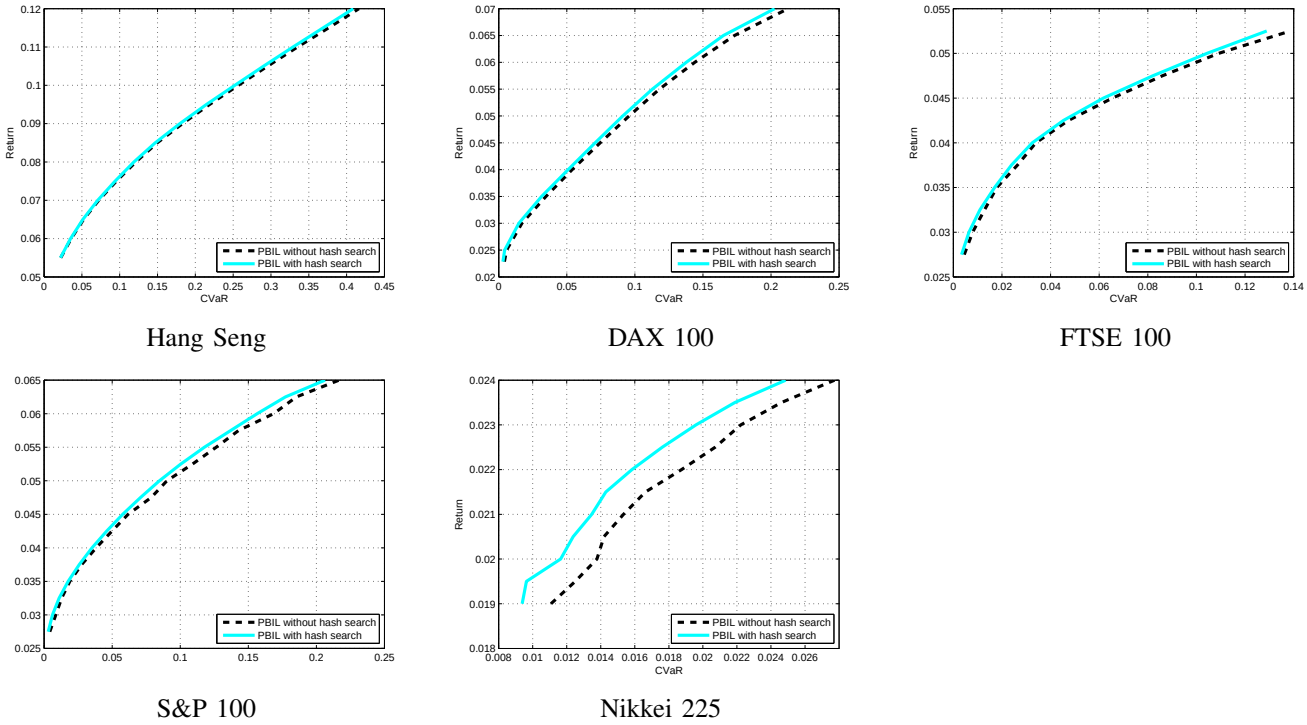


Fig. 3. Comparative results of PBIL with and without hash search for 5 general market instances.

Algorithm 4: Hash search

```
for each new individual h generated do
    for each key ky1 in HashtableInfeasibleSolution do
        if SimilarityCheck(h, ky1) ≥ K − 1 then
            re-generate individual h;
            break ;
    for each key ky2 in HashtableBadSolution do
        if SimilarityCheck(h, ky2) ≥ K − 2 then
            re-generate individual h;
            break ;
```

(which is used to control the kinds of pivots permitted) and the time allowed for each fitness calculation. That means we do not need to compute the optimal values for every individual. We only need to calculate the optimal value once for the global best solution after the search of the hybrid algorithm is finished. This will help improve the efficiency.

G. Local search

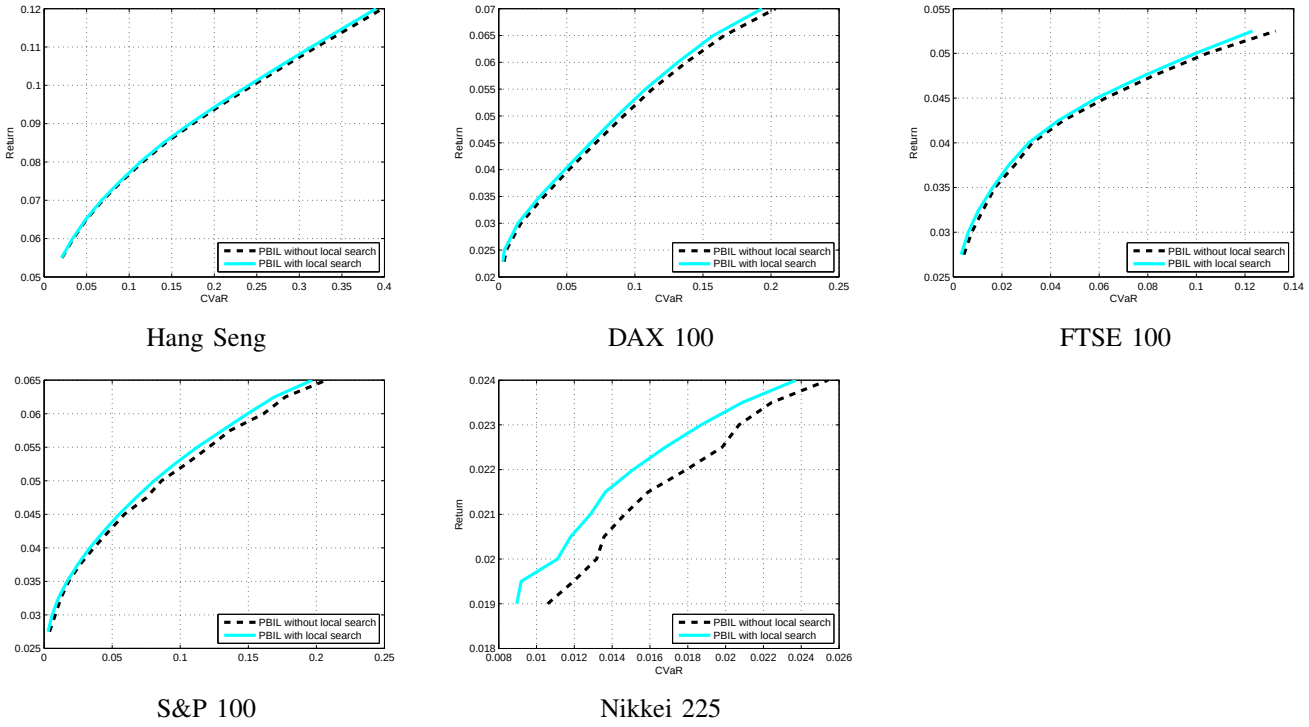
After the evaluation is done, top 20% individuals with the smallest fitness value of the generation are selected and the local search are applied to them in order to seek for the better solutions and evolve better individuals within a neighbourhood. Each time we replace one asset with a neighbourhood asset and then re-evaluate the new portfolio. The neighborhood relation of an asset is defined as the asset with the closest probability. If a better solution is obtained, the current best solution is updated. For each asset, we search na neighbours (i.e. na closest probability successors). The local search applied here is the incomplete neighbourhood search. It aims to seek for the possible improvement of the current solution. Figure 4 shows that the local search can indeed help the algorithm find better global solutions. The details of the local search are given as Algorithm 5.

Algorithm 5: Local search

```
Select the top 20% individuals of the generation;
for each selected individual do
    for i = 1 to K do
        for j = 1 to na do
            Replace asset i with a neighbourhood asset to
            form a new portfolio;
            Evaluate the new portfolio;
            if The fitness value of the new portfolio is
            smaller than the current best solution then
                Update the current best solution;
```

H. Archive

As mentioned in section IV-E, during the evolution, it is possible to obtain infeasible solutions. It's important to keep an archive of them. We use a hash table to record all the infeasible solutions obtained. Similarly we use a hash table



Hang Seng DAX 100 FTSE 100

S&P 100 Nikkei 225

Fig. 4. Comparative results of PBIL with and without local search for 5 general market instances.

to keep all bad solutions of every iteration. The purpose of maintaining the two hash tables is to avoid unnecessary search so that better solutions can be explored. We also keep a record of the best solutions obtained at each iteration to ensure that the good solutions found by the algorithm are not lost (i.e. elitist selection).

I. Update

In PBIL, the probability vector $\mathbf{v}$ can be considered as a prototype vector which is used to store the knowledge collected during the evaluation of current generation in order to guide the following population generations. $\mathbf{v}$ is updated by learning from the current best solution s_i^{cbest} and the current worst solution s_i^{cworst} using a positive learning rate lr and a negative learning rate $nelr$ correspondingly. Both the positive learning rate and the negative learning rate are used to control the speed of the prototype vector shifting to the better solution vector and the portions of exploration of the search space [61], [64]. The details of the probability updated are given as Algorithm 6.

Algorithm 6: Probability update

for *i = 1 to Q* **do**
  $v_i = v_i \times (1 - lr) + s_i^{cbest} \times lr$;
  if $s_i^{cbest} \neq s_i^{cworst}$ **then**
    $v_i = v_i \times (1 - nelr) + s_i^{cbest} \times nelr$;

J. Mutation

At the end of each iteration, the probability vector $\mathbf{v}$ is mutated according to a certain mutation probability mp. In this work, we use a mutation strategy, namely partially guided mutation [9]. It gives the equal chance to mutate $\mathbf{v}$ either randomly or based on the global best solution using a mutation rate mr. The advantage of doing this is that it can exploit the good structures in the current best solutions as well as exploring other regions of the search space at the same time. The details of the partially guided mutation are given as Algorithm 7.

Algorithm 7: Mutation (Partially guided mutation)

for *i = 1 to Q* **do**
  if $rand(0, 1] < mp$ **then**
    if $rand(0, 1] < 0.5$ **then**
      $r = Rand[0, 1]$;
      $v_i = v_i \times (1 - mr) + r \times mr$;
    else
      $v_i = s_i^{cbest}$;

V. Data Sets

In this work, we use the five benchmark instances which extend from the OR-library [65]. It contains 261 weekly historical price data for each asset of the following five different capital market indices:

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

where Q is the number of assets available for each market index. The weekly historical price data are used for generating the scenarios for the two-stage stochastic portfolio optimization model.

VI. Scenarios

A. Scenario generation

The scenarios need to be generated in order to represent the uncertain asset prices. The model has two stages. We use the price information on the recourse nodes to form the initial portfolio and use the price information on the successor evaluate nodes to perform the portfolio rebalancing actions. In this case, the 261 weekly historical price data from the OR-library cannot be used directly as it would lead to a prohibitively huge multi-stage problem. Instead, we have the following: we take the week 1 data $q_1^1, \ldots, q_1^Q$ as the initial price for the assets. Start from week 2, we compute the ratio between the price of the assets of two consecutive weeks $\Delta_t = q_{t+1}^i / q_t^i$ where $i = 1 \ldots Q$, $t = 1 \ldots 260$. Then we can obtain 260 new price data by computing $q_1^i * (1 + \Delta_t) \forall i \in Q, t = 1, \ldots, 260$.

Then we apply the copula scenario generation method [51] using the 260 new price data as the inputs to generate 400 recourse node scenarios. The evaluate nodes should be only dependent on their predecessor recourse node. Therefore, for each recourse node, we use the price scenario on that node and multiply a random coefficient within $(0.9, 1.1)$ to produce 40 corresponding scenarios for each of the evaluate nodes. The random coefficients are used to simulate the fluctuation of asset price in the second stage. We do not claim they are the optimal choices. There will be $400 \times 40 = 16000$ possibilities of scenarios in total and the evaluate scenarios are different for each different recourse node. By performing some experiments, we find the computational results are sensitive to the scenarios generated, especially for the evaluate node scenarios. Again, we do not claim the scenario generation methods we used are the best choices. Our primary aim is rather to develop an efficient method that can solve the two-stage stochastic portfolio optimization problem with a larger number of scenarios and to test the effectiveness of our hybrid combinatorial approach.

B. Stability

In stochastic programming, scenario generation methods are used to create a limited discrete distribution from the input data. The statistical properties of the scenario sets created should match the corresponding values estimated from the input data and the scenario generation method should not lay bias on the results by causing instability of the solutions. Usually, the stability tests are performed [66], [67] and there are two types of stability.

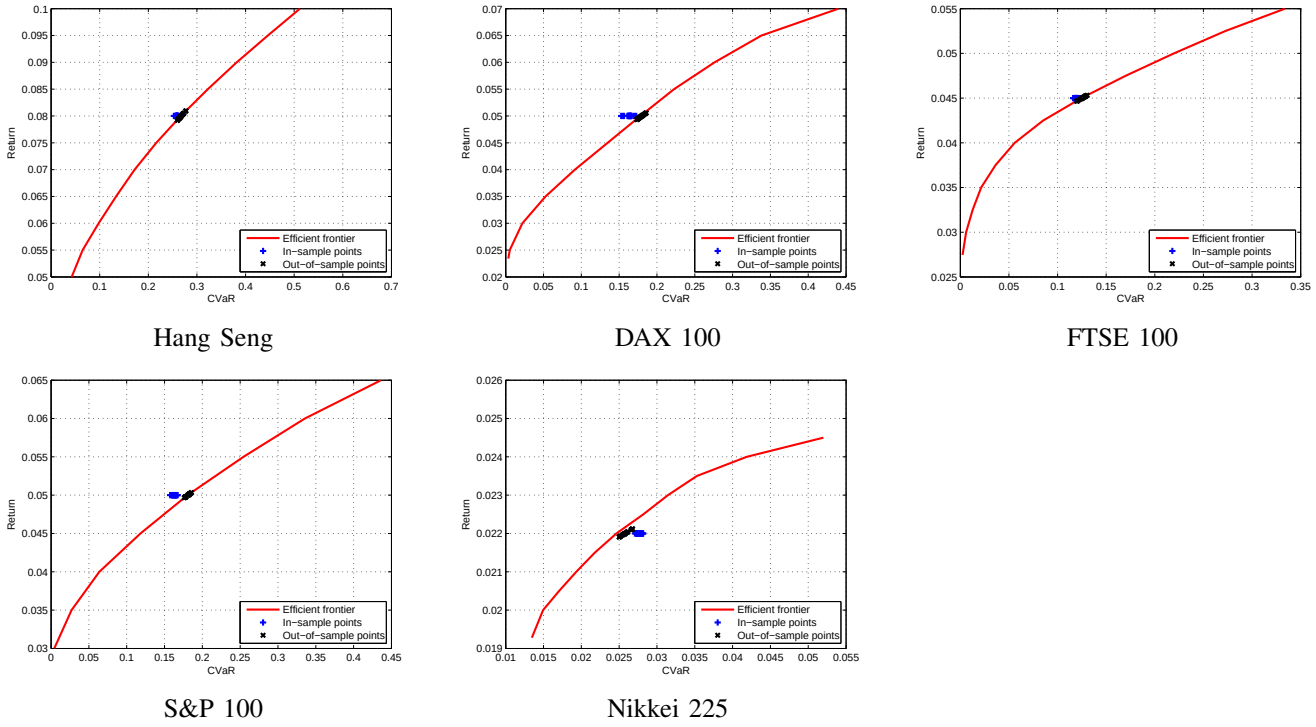
- *In-sample stability:* The scenario generation method is assessed in terms of its ability to match the benchmark distribution. We generate several scenario sets of a given size using the same input data. The idea is, no matter which scenario set we choose, the optimal objective value of the model should be approximately the same. The objective values should not vary across scenario sets. For this work, we use copula-based scenario generation method to generate 25 different scenario trees with the size 400 using the same input data. Then we use CPLEX to compute the optimal objective value of a same target return level for each scenario tree and compare the results. Ideally these results should be equal.
- *Out-of-sample stability:* The scenario generation method is assessed in terms of its ability to provide the stable results with respect to the benchmark distribution. We generate several scenario sets of a given size using the same input data and solve the model with each scenario set. The idea is, if we simulate the solutions obtained for each scenario set on the benchmark distribution, the value of the true objective function should be approximately the same. For this work, we use copula-based scenario generation method to generate 25 different scenario trees with the size 400 using the same input data and then use CPLEX to solve the model with a same target return level for each scenario tree. After that we simulate the solutions obtained for each scenario tree on the benchmark distribution to compute the true objective function. It is important that the benchmark distribution is not generated by the same method we are using and in our case, we use the input data directly as our benchmark distribution. Finally we compare the results, ideally these results should be equal, they should be also equal to the in-sample values (approximately).

For this work, the copula-based scenario generation method is only used to create the scenario sets for the recourse nodes. The scenarios for the evaluate nodes are dependent on their predecessor nodes and the random coefficients are also involved. Therefore, we only examine the stability tests on the single-stage model (i.e. without rebalancing actions). The purpose of performing the stability tests here is to show the copula-based scenario generation method will not influence the results and it is a suitable scenario generation method for this work.

The results are shown in Figure 5. Table III,IV calculate the mean value, the median value and the standard deviation of in-sample and out-of sample results respectively. The standard deviation of both in-sample and out-of-sample results are small, indicating that the scenario generation method we use is effective, in the sense that it will not cause instability in the solutions of the model.

VII. Parameter settings

The parameter settings used in this work are shown as follows:



Hang Seng DAX 100 FTSE 100

S&P 100 Nikkei 225

Fig. 5. Stability test results for 5 general market instances of the one-stage model using 400 scenarios.

TABLE III
IN-SAMPLE STABILITY TEST RESULTS FOR 5 GENERAL MARKET INSTANCES USING 400 SCENARIOS

Instance Index	Q	N_r	Mean(%)	Median(%)	stdev(%)
Hang Seng	31	400	25.8579	25.8947	0.2408
DAX 100	85	400	16.3507	16.4025	0.4873
FTSE 100	89	400	11.9221	11.9367	0.1805
S&P 100	98	400	16.2933	16.3385	0.2646
Nikkei 225	225	400	2.7621	2.7564	0.0321
Average					0.2411

TABLE IV
OUT-OF-SAMPLE STABILITY TEST RESULTS FOR 5 GENERAL MARKET INSTANCES USING 400 SCENARIOS

Instance Index	Q	N_r	Mean(%)	Median(%)	stdev(%)
Hang Seng	31	400	26.7367	26.6460	0.5159
DAX 100	85	400	17.8633	17.8262	0.3993
FTSE 100	89	400	12.5446	12.5495	0.3330
S&P 100	98	400	18.0863	18.1676	0.2857
Nikkei 225	225	400	2.5850	2.5751	0.0610
Average					0.3190

A. Model parameters

For each given target expected return μ, we set the critical percentile level of CVaR $\beta = 95\%$, fixed buying cost $\eta_b = 0.5$, variable buying cost $\rho_b = 0.5\%$, fixed selling cost $\eta_s = 0.5$, variable selling cost $\rho_s = 0.5\%$, cardinality $K = 10$, minimum holding position $w_{min} = 1\%$, minimum trading size $t_{min} = 0.1\%$. The initial portfolio only involves cash and we set the initial cash $h = 100000$. We assume the probability of each scenario is equal and therefore $p_j = 1/N_r$, $p_{(j,e)} = 1/N_e^j$.

B. Algorithmic parameters

We set population size $Po = 200$, the number of generations $Ge = 50$, mutation rate $mr = 0.05$, mutation probability $mp = 0.05$ and the number of neighbourhood assets $na = 15$.

The learning rates has a big effect on our hybrid algorithm. The algorithm will focus on searching using the information gained about the search space by using a larger learning rate, this is called exploitation. On the other hand, the algorithm will jump to other areas in the search space by using a lower learning rate, this is called the exploration. In order to choose the suitable learning rates, we test 4 different sets of learning rates and run a simple ranking test (if one set of learning rates obtains the best (minimum) CVaR value, we rank it as 1; if one set of learning rates obtains the second-best CVaR value, we rank it as 2 and so on). The results are shown in Table V.

TABLE V
AVERAGE RANKS OF THE HYBRID COMBINATORIAL ALGORITHM WITH DIFFERENT SETS OF LEARNING RATES FOR 5 GENERAL MARKET INSTANCES USING 16000 POSSIBILITIES OF SCENARIOS

Instance Index	Q	N_r	N_e^j		lr= $nelr$=	0.001 0.00075	0.01 0.0075	0.1 0.075	1 0.75
Hang Seng	31	400	40	aveRk		1.1429	1.1429	**1.0000**	**1.0000**
DAX 100	85	400	40	aveRk		1.4545	1.3636	**1.1818**	1.3636
FTSE 100	89	400	40	aveRk		1.6000	1.5000	**1.4000**	2.1000
S&P 100	98	400	40	aveRk		1.9286	**1.2143**	1.9286	2.1429
Nikkei 225	225	400	40	aveRk		**2.0000**	2.2500	2.7500	3.0000
Average				aveRk		1.6252	**1.4942**	1.6521	1.9213

There is always a trade-off between exploitation and exploration. In our context, exploitation refers to the ability of our hybrid algorithm to fully search the entire market instance while exploration refers to the ability of our hybrid algorithm to use the knowledge learned about the assets to narrow the future search. The higher the learning rates are set, the more areas of the instance will be searched. The lower the learning rates are set, the more exploration will take place. For this work, the search space of each market instance is different from each other. We can see from Table V that, the bigger learning rates have a better performance for the smaller instances. $lr = 1$, $nelr = 0.75$ and $lr = 0.1$, $nelr = 0.075$ have the best average rank for Hang Seng index (with $Q = 31$) and $lr = 0.1$, $nelr = 0.075$ has the best average rank for DAX 100 index (with $Q = 85$) and FTSE 100 index (with $Q = 89$). As the size of the instance increases, the smaller learning rates tend to perform better. $lr = 0.01$, $nelr = 0.0075$ has the best average rank for S&P 100 index (with $Q = 89$) and $lr = 0.001$, $nelr = 0.00075$ has the best average rank for Nikkei 225 index (with $Q = 225$).

As we discussed in section VIII-B before, the information gained from the previous return levels can be used in the next following return levels. Therefore we can set the lower learning rates in the first half of the return levels in order to have a better exploration and set the lower learning rates in the second half of the return levels in order to have a better exploitation. For Hang Seng, DAX 100 and FTSE 100 instances, we set the positive learning rate $lr = 0.1$, the negative learning rate $nelr = 0.075$ for the first 10 return levels and then change to $lr = 1$, $nelr = 0.75$ for the last 10 return levels. For S&P 100 and Nikkei 225 instances, we set the positive learning rate $lr = 0.001$, the negative learning rate $nelr = 0.00075$ for the first 10 return levels and then change to $lr = 0.01$, $nelr = 0.0075$ for the last 10 return levels.

Please note that as our main purpose is to test the effectiveness of our hybrid combinatorial approach, we do not claim these parameter settings are the optimal choices. Our primary aim is rather to develop an efficient method that can solve the two-stage stochastic portfolio optimization problem with a larger number of scenarios.

VIII. Experimental Results

A. Comparison of computational results for the five general benchmark instances

The main idea of our combinatorial approach is the decomposition of the two-stage stochastic model into two parts. The first part is to search for the selection of assets and the second part is to determine the corresponding weights of the selected assets. The first part is solved by using PBIL-based hybrid algorithm and the second part can be solved by a standard LP solver.

Considering the time limitation, we choose 20 equally spaced return levels and for each different return level, we run our hybrid algorithm to obtain a portfolio. The set of the portfolios obtained can form a frontier which represents the trade-offs between the expected return and the CVaR value which is a risk indicator.

In order to test the effectiveness of our proposed hybrid algorithm, we compare our results with three other different approaches. These three approaches use GA mutation only, PSO and random search for the first part respectively while the second part is solved by the same LP solver. The comparative results can be found in Figure 6.

We run each of different algorithms 10 times and take the simple ranking test. The final average ranks of the 5 different algorithms for 20 return levels are shown in Table VI.

TABLE VI
AVERAGE RANKS OF THE 4 DIFFERENT ALGORITHMS FOR 5 GENERAL MARKET INSTANCES USING 16000 POSSIBILITIES OF SCENARIOS

Instance Index	Q	N_r	N_e^j		Hybrid -Algorithm	GA -Mutation	PSO	Random -Search
Hang Seng	31	400	40	aveRk	**1.0000**	1.8571	3.1429	3.7857
DAX 100	85	400	40	aveRk	**1.0000**	1.9091	3.0000	4.0000
FTSE 100	89	400	40	aveRk	**1.0000**	2.0000	3.0000	4.0000
S&P 100	98	400	40	aveRk	**1.0000**	2.0000	3.0000	4.0000
Nikkei 225	225	400	40	aveRk	**1.0000**	2.0000	3.0000	4.0000
Average				aveRk	**1.0000**	1.9532	3.0286	3.9571

From Figure 6 and Table VI we can see that our hybrid combinatorial algorithm outperforms all the other 3 algorithms on all 5 instances.

B. Portfolio composition

As we mentioned previously, good solutions tend to have the similar structures. In fact, for this two-stage stochastic model, good solutions for two consecutive return levels also share some similarities. For each market instance, we run our hybrid algorithm for 10 equally spaced return levels to obtain the assets selections. The results are shown in Figure 7. We calculate the average similarities for two consecutive return levels and the results are shown in Table VII. We can see that for two consecutive return levels, there are approximately 6.53 out of 10 identical asset choices on average.

TABLE VII
AVERAGE SIMILARITIES FOR TWO CONSECUTIVE RETURN LEVELS OF 5 GENERAL MARKET INSTANCES USING 16000 POSSIBILITIES OF SCENARIOS

Instance Index	Q	N_r	N_e^j	Average similarities
Hang Seng	31	400	40	6.78
DAX 100	85	400	40	6.11
FTSE 100	89	400	40	6.78
S&P 100	98	400	40	6.11
Nikkei 225	225	400	40	6.89
Average				6.53

This is a useful observation which can be used to guide our search. The idea is, we keep the best solution of one return level and use it as the starting search point of the next return level. This mechanism can be adopted in all 5 algorithms mentioned in section VIII-A. Another important component in our hybrid algorithm, the probability vector, also contains

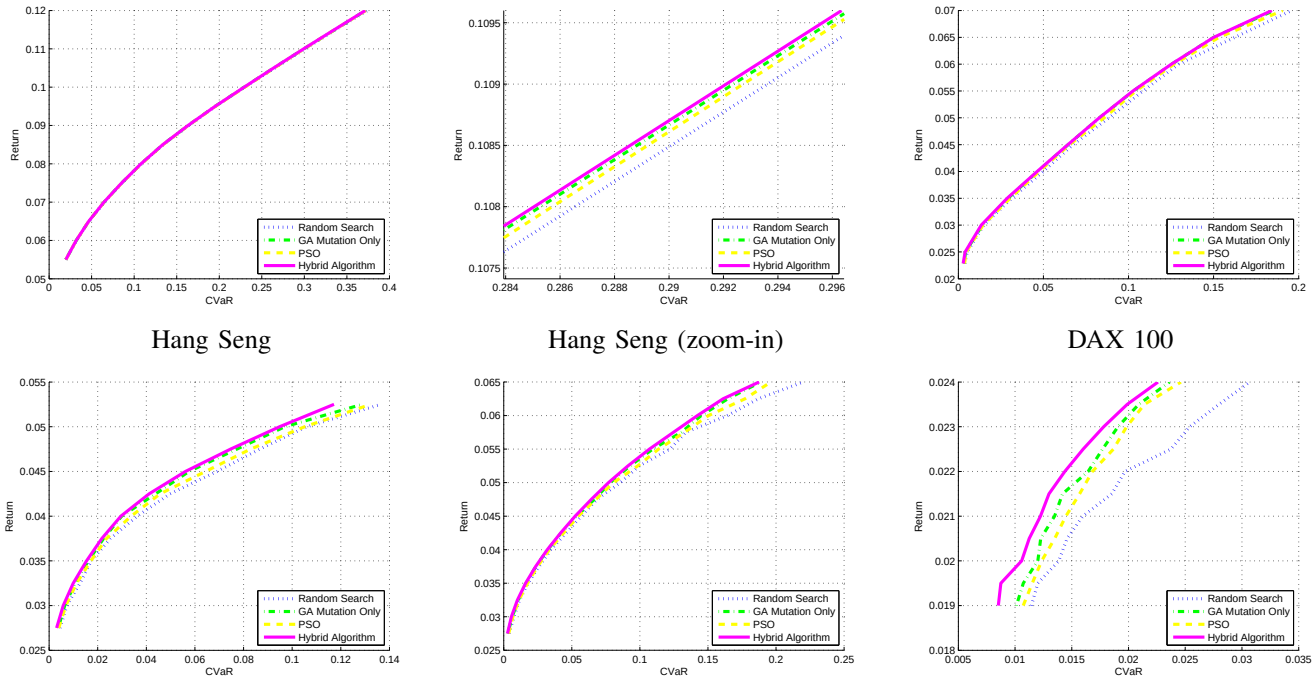


Fig. 6. Comparative results of the hybrid algorithm with 3 other different approaches for 5 general market instances using 16000 possibilities of scenarios.

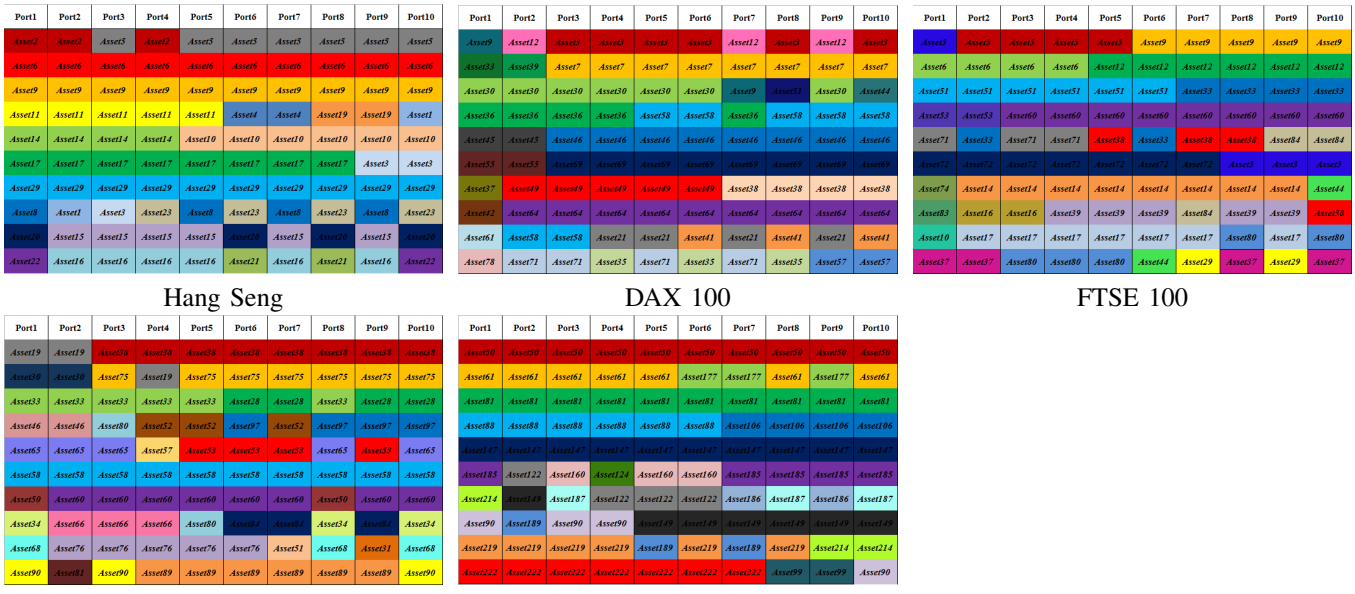


Fig. 7. Portfolio composition results of our hybrid algorithm for 5 general market instances using 16000 possibilities of scenarios. Each column represents the assets selection of the best portfolio obtained for one specified return level. One portfolio is composed of 10 different assets which are represented by 10 different color sectors. The same asset is represented by the same color sector in each market instance.

useful information. Derived from the ideas used in competitive learning, the whole population is defined as the probability vector representation. Therefore the good asset tends to have a high probability to be selected. The knowledge learned in one return level can be transferred to the next return level. The probability vector is adjusted accordingly in each generation and in each return level and it is gradually shifted towards representing better solutions.

C. Performance

All the algorithms for the two-stage stochastic portfolio optimization model mentioned in section VIII-A were implemented in C# with concert technology in CPLEX on top of CPLEX 12.4 solver. All the tests were run on the same Intel(R) Core(TM) i7-4600M 2.90GHz processor with 16.00 GB RAM PC and Windows 7 operating system. For a given return level of each different market instance, the computational time is given in table VIII.

TABLE VIII
COMPUTATIONAL TIME OF THE 4 DIFFERENT ALGORITHMS FOR 5 GENERAL MARKET INSTANCES USING 16000 POSSIBILITIES OF SCENARIOS

Instance Index	Q	N_r	N_e^j		Hybrid -Algorithm	GA -Mutation	PSO	Random -Search
Hang Seng	31	400	40	min	17	15	15	11
DAX 100	85	400	40	min	31	30	30	23
FTSE 100	89	400	40	min	32	30	31	23
S&P 100	98	400	40	min	35	34	34	24
Nikkei 225	225	400	40	min	56	54	54	45
Average				min	34.2	32.6	32.8	25.2

In order to conduct fair comparisons between the algorithms, all the tests were run under the same condition (i.e. the same number of generations). The performance of 4 algorithms are shown in Figure 8. As we can see that uur hybrid algorithm converges within less than 50 generations for all 5 market instances while the other 3 algorithms fail to converge within 100 generations. In fact, our hybrid algorithm can achieve better results with less time compared to the other 4 algorithms. According to the No-Free-Lunch theorem [68], there is no best optimization algorithm for all possible problems. The best algorithm for one problem should be specifically designed for that problem. For this work, we implement a model-specific hybrid combinatorial algorithm for the two-stage stochastic portfolio optimization problem. The hybrid algorithm is based on PBIL which has an important component, the probability vector. It enables learning during the whole execution in the sense that the knowledge from the previous return levels can be inherited to problems with the similar return levels.

A concern is that, the local search adopted in our hybrid algorithm, can be also adopted in GA with mutation only and PSO. The idea is, the probability vector in our hybrid algorithm can provide a more meaningful neighbourhood structure, therefore the local search are much more effective compared to using a random neighbourhood structure (i.e. replace an asset with a random one).

IX. CONCLUSION AND FURTHER WORK

In this work, we investigate a two-stage stochastic portfolio optimization model which minimizes the Conditional Value at Risk (CVaR) of the portfolio loss with a comprehensive set of real world trading constraints. The two-stage stochastic model can capture the market uncertainty in terms of future asset prices therefore it enables the investors rebalancing the assets. Stability tests are performed and the results confirm that the scenario generation method used for this work is effective.

A model-specific hybrid combinatorial approach is proposed for the two-stage stochastic model. It integrates a hybrid algorithm and an LP solver in the sense that hybrid algorithm can search for the assets selection heuristically while the LP solver can solve the corresponding reduced sub-problems optimally. The hybrid algorithm for assets searching is based on PBIL while local search and hash search are adopted in order to solve the two-stage stochastic model with a larger number of scenarios effectively and efficiently. Elitist selection and partially guided mutation are also adopted in order to enhance the evolution.

Comparison results against the other 3 algorithms are given for 5 general market instances and our hybrid combinatorial approach outperforms the 3 algorithms on all instances. We also investigate the structure of the solutions obtained and we demonstrate that the knowledge learned in one return level can be inherited to the next following return levels. This can enhance the search process and makes the whole execution more efficient. The effects of different learning rates are also examined in order to choose for the better settings for the hybrid algorithm.

This work is mainly focus on the algorithmic part. The scenarios which represent the possible future asset prices are highly depend on the historical data. In order to make the model more practical, the alternative representation of the possible future asset prices might be required. This can be the possible future research direction.

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, 1995.

[4] R. Mansini and M. G. Speranza, "Heuristic algorithms for the portfolio selection problem with minimum transaction lots," *European Journal of Operational Research*, vol. 114, no. 2, pp. 219 – 233, 1999.

[5] T. Cui, S. Cheng, and R. Bai, "A combinatorial algorithm for the cardinality constrained portfolio optimization problem," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, July 2014, pp. 491–498.

[6] 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.

[7] 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.

[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.

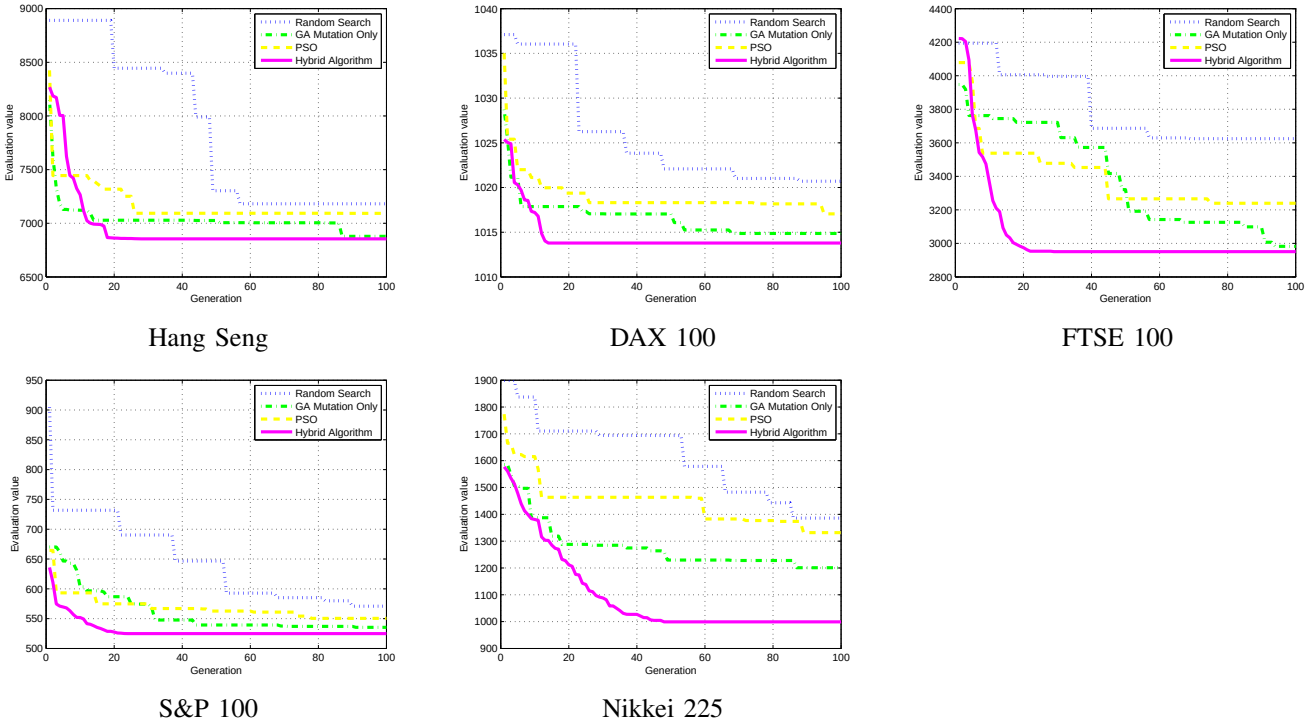


Fig. 8. The performance of 4 different algorithms for 5 general market instances using 16000 possibilities of scenarios.

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

[10] R. Baldacci, M. Boschetti, N. Christofides, and S. Christofides, "Exact methods for large-scale multi-period financial planning problems," *Computational Management Science*, vol. 6, no. 3, pp. 281–306, 2009.

[11] D. Barro and E. Canestrelli, "Dynamic portfolio optimization: Time decomposition using the maximum principle with a scenario approach," *European Journal of Operational Research*, vol. 163, no. 1, pp. 217–229, 2005, financial Modelling and Risk Management.

[12] J. Li and J. Xu, "Multi-objective portfolio selection model with fuzzy random returns and a compromise approach-based genetic algorithm," *Information Sciences*, vol. 220, no. 0, pp. 507 – 521, 2013, online Fuzzy Machine Learning and Data Mining.

[13] H. Yano, "Fuzzy decision making for fuzzy random multiobjective linear programming problems with variance covariance matrices," *Information Sciences*, vol. 272, no. 0, pp. 111 – 125, 2014.

[14] P. Gupta, M. Inuiguchi, M. K. Mehlawat, and G. Mittal, "Multiobjective credibilistic portfolio selection model with fuzzy chance-constraints," *Information Sciences*, vol. 229, pp. 1 – 17, 2013.

[15] P. Gupta, M. K. Mehlawat, and A. Saxena, "A hybrid approach to asset allocation with simultaneous consideration of suitability and optimality," *Information Sciences*, vol. 180, no. 11, pp. 2264 – 2285, 2010.

[16] P. Kall and S. Wallace, *Stochastic programming*, ser. Wiley-Interscience series in systems and optimization. Wiley, 1994.

[17] J. Birge and F. Louveaux, *Introduction to Stochastic Programming*, ser. Springer Series in Operations Research and Financial Engineering. U.S. Government Printing Office, 1997.

[18] G. Dantzig, *Linear programming and extensions*, ser. Rand Corporation Research Study. Princeton, NJ: Princeton Univ. Press, 1963.

[19] G. B. Dantzig, "Linear programming under uncertainty," *Management Science*, vol. 50, no. 12 Supplement, pp. 1764–1769, Dec. 2004.

[20] A. J. King and S. W. Wallace, *Modeling with Stochastic Programming*, ser. Springer Series in Operations Research and Financial Engineering. Springer New York, 2012.

[21] S. W. Wallace and W. T. Ziemba, *Applications of Stochastic Programming*, 1st ed. Society for Industrial and Applied Mathematics, June 2005.

[22] R. Bai, S. W. Wallace, J. Li, and A. Y.-L. Chong, "Stochastic service network design with rerouting," *Transportation Research Part B: Methodological*, vol. 60, no. 0, pp. 50 – 65, 2014.

[23] P. Jorion, *Value at Risk: The New Benchmark for Managing Financial Risk*, 3rd ed. McGraw-Hill Education, 2006.

[24] M. Pritsker, "Evaluating Value at Risk Methodologies: Accuracy versus Computational Time," *Journal of Financial Services Research*, vol. 12, no. 2, pp. 201–242, October 1997.

[25] *RiskMetrics technical manual*, 4th ed., J.P. Morgan, New York, 1995.

[26] J. W. Goh, K. G. Lim, M. Sim, and W. Zhang, "Portfolio value-at-risk optimization for asymmetrically distributed asset returns," *European Journal of Operational Research*, vol. 221, no. 2, pp. 397 – 406, 2012.

[27] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical Finance*, vol. 9, no. 3, pp. 203–228, 1999.

[28] H. Mausser and D. Rosen, "Beyond var: from measuring risk to managing risk," in *Proceedings of the IEEE/IAFE 1999 Conference on Computational Intelligence for Financial Engineering (CIFEr 1999)*, 1999, pp. 163–178.

[29] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *Journal of Risk*, vol. 2, pp. 21–41, 2000.

[30] B. Golub, M. Holmer, R. McKendall, L. Pohlman, and S. A. Zenios, "A stochastic programming model for money management," *European Journal of Operational Research*, vol. 85, no. 2, pp. 282 – 296, 1995.

[31] C. Vassiadou-Zeniou and S. A. Zenios, "Robust optimization models for managing callable bond portfolios," *European Journal of Operational Research*, vol. 91, no. 2, pp. 264 – 273, 1996.

[32] A. Beltratti, A. Consiglio, and S. Zenios, "Scenario modeling for the management of international bond portfolios," *Annals of Operations Research*, vol. 85, no. 0, pp. 227–247, 1999.

[33] J. M. Mulvey and H. Vladimirou, "Stochastic network programming for financial planning problems," *Manage. Sci.*, vol. 38, no. 11, pp. 1642–1664, Nov. 1992.

[34] J. M. Mulvey and W. T. Ziemba, *Handbooks in Operations Research and Management Science*. Elsevier, 1995, vol. Volume 9, ch. Chapter 15 Asset and liability allocation in a global environment, pp. 435–463.

[35] J. M. Mulvey, D. P. Rosenbaum, and B. Shetty, "Parameter estimation in stochastic scenario generation systems," *European Journal of Operational Research*, vol. 118, no. 3, pp. 563 – 577, 1999.

[36] N. Topaloglou, H. Vladimirou, and S. A. Zenios, "A dynamic stochastic

programming model for international portfolio management," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1501–1524, 2008.

[37] S. J. Stoyan and R. H. Kwon, "A stochastic-goal mixed-integer programming approach for integrated stock and bond portfolio optimization," *Computers & Industrial Engineering*, vol. 61, no. 4, pp. 1285 – 1295, 2011.

[38] L.-Y. Yu, X.-D. Ji, and S.-Y. Wang, "Stochastic programming models in financial optimization: A survey," *Advanced Modeling and Optimization*, vol. 5, no. 1, 2003.

[39] S. J. Stoyan and R. H. Kwon, "A two-stage stochastic mixed-integer programming approach to the index tracking problem," *Optimization and Engineering*, vol. 11, no. 2, pp. 247–275, 2010.

[40] S.-E. Fleten, K. Hyland, and S. W. Wallace, "The performance of stochastic dynamic and fixed mix portfolio models," *European Journal of Operational Research*, vol. 140, no. 1, pp. 37 – 49, 2002.

[41] J. L. Higle and S. W. Wallace, "Sensitivity analysis and uncertainty in linear programming," *Interfaces*, vol. 33, no. 4, pp. 53–60, 2003.

[42] D. Barro and E. Canestrelli, "Dynamic portfolio optimization: Time decomposition using the maximum principle with a scenario approach," *European Journal of Operational Research*, vol. 163, no. 1, pp. 217 – 229, 2005, financial Modelling and Risk Management.

[43] A. A. Gaivoronski, S. Krylov, and N. van der Wijst, "Optimal portfolio selection and dynamic benchmark tracking," *European Journal of Operational Research*, vol. 163, no. 1, pp. 115 – 131, 2005, financial Modelling and Risk Management.

[44] L. Escudero, A. Garn, M. Merino, and G. Prez, "A two-stage stochastic integer programming approach as a mixture of branch-and-fix coordination and benders decomposition schemes," *Annals of Operations Research*, vol. 152, no. 1, pp. 395–420, 2007.

[45] S. Greco, B. Matarazzo, and R. Sowiski, "Beyond markowitz with multiple criteria decision aiding," *Journal of Business Economics*, vol. 83, no. 1, pp. 29–60, 2013.

[46] Y. Chen and X. Wang, "A hybrid stock trading system using genetic network programming and mean conditional value-at-risk," *European Journal of Operational Research*, vol. 240, no. 3, pp. 861 – 871, 2015.

[47] L. Yu, S. Wang, Y. Wu, and K. Lai, "A dynamic stochastic programming model for bond portfolio management," in *Computational Science - ICCS 2004*, ser. Lecture Notes in Computer Science, M. Bubak, G. van Albada, P. Sloot, and J. Dongarra, Eds. Springer Berlin Heidelberg, 2004, vol. 3039, pp. 876–883.

[48] F. He and R. Qu, "A two-stage stochastic mixed-integer program modelling and hybrid solution approach to portfolio selection problems," *Information Sciences*, vol. 289, no. 0, pp. 190 – 205, 2014.

[49] M. Rysz, A. Vinel, P. Krokhmal, and P. Eduardo, "A scenario decomposition algorithm for stochastic programming problems with a class of downside risk measures," *INFORMS Journal on Computing*, vol. 27, no. 2, pp. 416 – 430, 2015.

[50] T. Cui, R. Bai, A. J. Parkes, F. He, R. Qu, and J. Li, "A hybrid genetic algorithm for a two-stage stochastic portfolio optimization with uncertain asset prices," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*, May 2015.

[51] M. Kaut and S. W. Wallace, "Shape-based scenario generation using copulas," *Computational Management Science*, vol. 8, no. 1–2, pp. 181–199, 2011.

[52] G. C. Pflug, "Some remarks on the value-at-risk and the conditional value-at-risk," in *Probabilistic Constrained Optimization*. Springer US, 2000, vol. 49, pp. 272–281.

[53] S. Uryasev, "Introduction to the theory of probabilistic functions and percentiles (value-at-risk)," in *Probabilistic Constrained Optimization*, ser. Nonconvex Optimization and Its Applications, S. Uryasev, Ed. Springer US, 2000, vol. 49, pp. 1–25.

[54] K. Høyland and S. W. Wallace, "Generating scenario trees for multistage decision problems," *Manage. Sci.*, vol. 47, no. 2, pp. 295–307, Feb. 2001.

[55] K. Høyland, M. Kaut, and S. W. Wallace, "A heuristic for moment-matching scenario generation," *Computational Optimization and Applications*, vol. 24, no. 2–3, pp. 169–185, 2003.

[56] F. Andersson, S. Uryasev, and S. Uryasev, "Credit risk optimization with conditional value-at-risk criterion," *Mathematical Programming*, vol. 89, no. 2, pp. 273–291, January 2001.

[57] P. Krokhmal, J. Palmquist, and S. Uryasev, "Portfolio optimization with conditional value-at-risk objective and constraints," *Journal of Risk*, vol. 4, pp. 11–27, 2002.

[58] M. Woodside-Oriakhi, C. Lucas, and J. Beasley, "Portfolio rebalancing with an investment horizon and transaction costs," *Omega*, vol. 41, no. 2, pp. 406 – 420, 2013, management science and environmental issues.

[59] M. Dyer and L. Stougie, "Computational complexity of stochastic programming problems," *Mathematical Programming*, vol. 106, no. 3, pp. 423–432, May 2006.

[60] A. Shapiro and A. Nemirovski, "On complexity of stochastic programming problems," *Continous Optimization*, pp. 111–146, 2004.

[61] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Pittsburgh, PA, USA, Tech. Rep., 1994.

[62] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm," Pittsburgh, PA, USA, Tech. Rep., 1995.

[63] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[64] J. L. Shapiro, "The sensitivity of PBIL to its learning rate, and how detailed balance can remove it," in *Proceedings of the Seventh Workshop on Foundations of Genetic Algorithms, Torremolinos, Spain, September 2-4, 2002*, 2002, pp. 115–132.

[65] 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.

[66] M. Kaut, S. W. Wallace, H. Vladimirou, and S. Zenios, "Stability analysis of portfolio management with conditional value-at-risk," *Quantitative Finance*, vol. 7, no. 4, pp. 397–409, 2007.

[67] M. Kaut and S. W. Wallace, "Evaluation of scenario-generation methods for stochastic programming," *Pacific Journal of Optimization*, vol. 3, no. 2, pp. 257–271, 2007.

[68] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 67–82, Apr 1997.