Swarm Intelligent Optimisation based Stochastic Programming Model for Dynamic Asset Allocation

Jing Dang, David Edelman, Ronald Hochreiter and Anthony Brabazon

Abstract— **Asset allocation is critical for the portfolio management process. In this paper, we solve a dynamic asset allocation problem through a multiperiod stochastic programming model. The objective is to maximise the expected utility of wealth at the end of the planning periods. To improve the optimisation result of the model, we employ swarm intelligent optimisers, the Bacterial Foraging Optimisation (BFO) algorithm and the Particle Swarm Optimisation (PSO) algorithm. A hybrid optimiser using the Bacterial Foraging Optimisation algorithm for initialisation and the Sequential Quadratic Programming (SQP) for local search is also suggested. The results are compared with the standard-alone SQP and the canonical Genetic Algorithm. The numerical results suggest the hybrid method provides better result, with improved accuracy, stability and computing speed than using BFO, PSO, GA, or SQP alone.**

I. Introduction

Asset allocation is the selection of a portfolio of investments, where each component is an asset class rather than an individual security. Most portfolios contain risky assets. Fluctuations in the values of such risky assets will generally cause the value of the portfolio to change. Investors must decide how to rebalance the portfolio, making asset allocation decisions in response to such changes. Dynamic asset allocation strategies are explicit rules of doing so. Asset allocation decisions are critical for investors with diversified portfolios. Institutional investors must manage their strategic asset mix over time to achieve favorable returns, in presence of uncertainties and subject to various legal constraints, policies, and other requirements. A classical approach used for asset allocation is the Markowitz mean-variance model [13]. But it suffers from several deficiencies as a single-period model [14]. In this paper, we consider a multiperiod portfolio optimisation model to determine the optimal asset mix over time.

Stochastic Programming (SP) method (see [1] for an overview of the method) is one of the major approaches solving practical dynamic asset allocation problems. The aim of SP model is to find an optimal decision in problems involving uncertain data. In this terminology, *stochastic* is opposed to *deterministic* and means that some data are random, whereas programming refers to the fact that various parts of the problem can be modelled as linear or nonlinear mathematical programs. The field, also known as *optimisation under uncertainty*, is developing rapidly with contributions from many disciplines such as operations research, economics and finance, mathematics, probability and statistics [16]. SP model usually takes the format of a scenario tree. Scenarios represent the range of possible outcomes for the uncertainties. Rather than finding a generic optimal policy, the optimal solution is computed at each node in the scenario tree, for the specified decision variables.

In this paper, we apply the multiperiod stochastic programming model to an asset allocation problem as follows: *Given a set of assets, a fixed planning horizon and a set of rebalance dates, find the strategic asset allocation strategy that maximises the utility subject to the constraints.* Given the problem setup (investment horizon, rebalancing schedule, number of scenarios and risk budget, etc.), we generate a scenario based decision tree and find the optimal active investment strategy correspondingly. The Stochastic Programming model allows for a wide range of objectives, legal and policy constraints and risk measures. It has been applied to solve asset and wealth management problems for both large financial insitutions and individuals [20].

Existing stochastic programming algorithmic systems and applications are introduced in [19]. To improve the optimisation result of the SP model for our financial problem, we apply swarm intelligent optimisers. As they exhibit flexible, robust, self-organized behavior, and hence used to solve complex tasks [11]. We employ two main optimisers in the field, the Bacterial Foraging Optimisation (BFO) algorithm and the Particle Swarm Optimisation (PSO) algorithm. BFO models the foraging behavior of *E. coli* bacteria as an optimisation process, and the PSO algorithm is inspired by the flocking and schooling behaviour of birds and fish. A hybrid optimiser using the Bacterial Foraging Optimisation algorithm for initialisation and the Sequential Quadratic Programming (SQP) for local search is also suggested.

The rest of the paper is organized as follows: Section II illustrates the asset allocation problem this paper aims to solve. Section III provides a concise overview of the multiperiod stochastic programming method, where Section III-A outlines detailed procedure of the scenario generation step, and Section III-B describes the swarm intelligent optimisers employed. Section IV provides numerical results followed by conclusions and future work in Section V.

II. Asset Allocation Problem

In the following we illustrate a simplified investment strategy without consideration of the transaction cost. An

Jing Dang, David Edelman and Anthony Brabazon are from the Natural Computing Research and Applications Group, and School of Business, University College Dublin, Ireland; (email: {jing.dang, davide, anthony.brabazon}@ucd.ie)

Ronald Hochreiter is with the Department of Statistics and Mathematics, WU Viena University of Economics and Business; (email: ronald.hochreiter@wc.ac.at)

active manager manages a portfolio of two instruments, a 'safe' money market index and a 'risky' equity index, against a benchmark of the money market index (e.g., the simulated LIBOR rate), rebalancing the strategy periodically. We mimic the manager's decision process within a multiperiod stochastic programming framework, where the asset returns are simulated by the correlated Geometric Brownian Motions (GBM) with a drift equal to the historical risk premium. We try to find the optimal allocations between the two assets using an objective function so that the distribution of the gap between the final Net Asset Value (NAV) and the benchmark across all scenarios reflects the risk budget assigned by the client.

Let Δt be the simulation time step, T be the horizon of the planning period. Then the set of simulation time periods is $t \in \mathbb{T} := \{0, \Delta t, 2\Delta t, T\}$, where time 0 is the current time. Note, by time t, we mean the span of time period $(t-1, t]$ Let $\mathbb{S}$ be the set of all simulated scenarios, $s \in \mathbb{S} := \{s_t | t \in \mathbb{T}\}$. And let I be the set of all assets, $i \in I$. For each $i \in I$, $t \in T$, and $s \in S$, we define the following parameters and decision variables.

Parameters:

- $V_{i,t}^s$ The value (price) for each asset i, at each time t, and for each scenario s, where $i \in I, t \in T, s \in S$.
- L_t^s The value of the liability (e.g. LIBOR rate) at time t and in each scenario s. For example, if we set $L_t^s = V_{i,t}^s$, then it means our benchmark is one of the asset classes the portfolio can invest in.
- W_t Wealth at the beginning of time period t, where $W_t = \sum_{s \in S} \pi_t^s W_t^s$.
- π_t^s The probability that scenario s occurs at time t, where $\sum_{s \in S} \pi_t^s = 1$

Decision variables:

- $\theta_{i,t}^s$ Relative portfolio weight of asset i, at the time t, under scenario s

The optimal allocation strategy is determined by maximising the expected utility of the wealth process, subject to the constraint structure. In our model set up, we employ the semivariance utility function as suggested in [17] (also mentioned in [4] as the downside-quadratic utility function). The general form of the period utility function is:

$$u_t(W_t) = (1-\beta)W_t - \beta\left((W_t - L_t)_-\right)^2$$

where $(W_t - L_t)_-$ is the loss function and defined as $\max(0, L_t - W_t)$, L denotes the target wealth, $\beta \in [0, 1]$ is the risk preference parameter. As mentioned in [17], this utility function displays constant relative risk aversion, meaning that the relative allocation to risky assets does not change as wealth increases on the upside ($W > L$). And the utility function displays decreasing relative risk aversion on the downside ($W < L$), indicating that the allocation to risky assets will decreases as the wealth decreases leading to a truncated left tail in the expected final wealth distribution. The degree of downside risk aversion is determined by the value of β. β close to 1 denotes heavily risk averse and severely punishes scenarios in which there is a shortfall, β with a value of 0 represents pure wealth optimisation). The expected utility of the wealth process is:

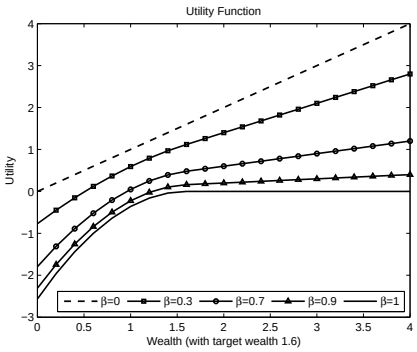


Fig. 1. Downside-quadratic utility function)

$$\begin{aligned} U &= \sum_{s \in \mathbb{S}} \pi^s \sum_{t \in \mathbb{T}} u_t(W_t^s) \\ &= \sum_{s \in \mathbb{S}} \pi^s \sum_{t \in \mathbb{T}} \left[(1-\beta)W_t^s - \beta\left((W_t^s - L_t^s)_-\right)^2\right] \end{aligned}$$

Based on the above, the deterministic equivalent representation of the resulting multiperiod stochastic program is formulated as below, where we use a minimisation form of the objective function in accordance with our global optimiser:

Model:

$$\text{Minimise} \quad U_T = \sum_{s \in \mathbb{S}} \pi^s \left[\beta\left((W_T^s - L_T^s)_-\right)^2 - (1-\beta)W_T^s\right]$$

subject to

$$\sum_{i \in I} \theta_{i,t}^s = 1, \quad \forall\, t \in T \text{(weight constraint)}$$

$$\theta_{i,t}^s \geqslant 0, \quad \forall\, t \in T,\ s \in S \text{(no short selling constraint)}$$

The portfolio wealth at each time t and each scenario s is denoted by

$$W_t^s = W_{t-1}^s \textstyle\sum_{i \in I} \theta_{i,t-1}^s (1 + r_{i,t}^s)$$

where $W_0 = 1$(million) and $r_{i,t}^s = V_{i,t}^s / V_{i,t-1}^s - 1$ denotes the return on each asset for a given time frame in a given scenario. In this model, the objective utility function is binding at the simulation terminal nodes, alternatives could be considered that binding the utility functions at the simulated intermediary nodes.

III. Stochastic Programming Framework

Stochastic Programming method is one of the major approaches solving practical dynamic asset allocation problems. In this section, we introduce two major process of the SP method. The first is the scenario generation, and the second is the optimsation process.

A. Scenario Generation

The concept of scenarios is usually employed for the modelling of randomness in stochastic programming models ([5]), in which data evolve over time and decisions have to be made independently upon knowing the actual paths that will occur. The data are usually subject to uncertainty or some kind of risk. In the asset allocation problem, the return values of each asset are the uncertain variables and investment decisions must be made before the actual asset performances can be observed. Each scenario can be viewed as one possible realisation of the underlying uncertainty, which could be modelled through the multivariate stochastic process. In general, two approaches are available for the design of a multi-stage scenario generator [8]:

- Direct scenario tree sampling. Based on historical data or econometric model use sampling or node-wise approximation methods to build the scenario tree iteratively from the root node to the terminal stage.
- Scenario path simulation and optimal tree approximation. Use pre-sampled scenario paths to build an optimal approximation of this data set.

In this paper, we use the latter approach due to their advantage in describing uncertainties with more accuracy [9]. The detailed steps are illustrated in the following.

1) Path Simulation: To generate simulated sample paths, we follow the procedures as below:

- Choosing the appropriate data generating process, i.e. the correlated Geometric Brownian Motion (GBM) process
- Calibrating the model based on historical data
- Generating data paths across the planning horizon

In our model, asset value $V^s_{i,t+\Delta t}$ is generated by Monte Carlo simulation based on the multivariate (correlated) Geometric Brownian Motion (GBM) process [7]:

$$V^s_{i,t+\Delta t} = V^s_{i,t} exp\left((\mu_i - \frac{1}{2}\sigma_i^2)\Delta t + \sqrt{\Delta t}\sum_{j=1}^{n} A_{ij} Z_{t,j}\right)$$

where n denotes the dimension of the assets. $Z_t = (Z_{t,1}, ..., Z_{t,n}) \sim N(0, I)$ and $Z_1, Z_2, ..., Z_t, ...$ are independent, A is be the Cholesky factor of the covariance of assets.

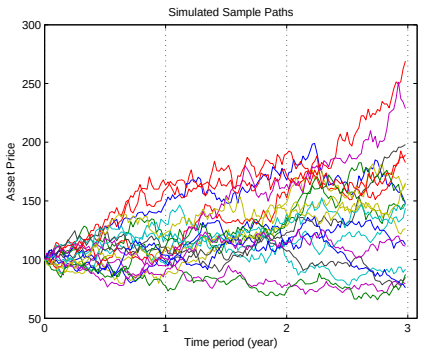


Fig. 2. Path Simulation

Figure 2 illustrates possible results of generating 20 simulated sample paths. In this paper, we consider paths simulated with equal probability, our future work would incorporate consideration of extreme events, involving some probability matching techniques. We describe the decision tree construction procedure based on this in the following.

2) Constructing Decision Tree: To generate the decision tree based on the simulated sample paths over the planning period, we employ the *forward sequential clustering algorithm*. Basically, in each period sequentially, a chosen clustering technique is applied to the data set of simulated paths. In our implementation, we apply *K-means clustering* technique, which is a randomised algorithm. The clustering techniques are implemented based on similarities calculated from distances between sampled return vectors. It partitions n samples into k clusters, by minimising the sum of the squared distances to the cluster centers [12]. The detailed procedure is illustrated in Algorithm 1 below.

Initialising the structure of the scenario tree: the number of stages T and the branching scheme at each stage N_t.

while $t < T$ **do**
  if $t == 1$ **then**
    Applying clustering technique to the sample data points at time t (based on the Euclidean distances between sampled return vectors).
    Recording the clustering index vector I_t.
  else
    for $i = 1 : N_{t-1}$ **do**
      Applying clustering technique to the sample data points at time t, which branched from the same centorid i in the last stage $t-1$
      Recording the clustering sub index vector $I'_{t,i}$
    end
    Updating the clustering index vector I_t
    t = t+1
  end
end

Algorithm 1: Forward sequential clustering algorithm

The decisions are made conditionally under the framework using simulated paths. At each stage, several clusters of simulated paths are generated, and the decision tree is constructed where conditional decisions are made at each nodes. We illustrate this procedure use a simple case of 20 simulated paths over three periods, with clustering results shown in Figure 3. For simplicity, assuming a branching factor of $[1, 3, 2]$, which implies that 3 clusters of 20 paths are made at $t = 1$, and 2 more cluster corresponding to each stage 1 cluster block is made at $t = 2$. The decision tree would result in 10 (1+3+6) decision nodes where corresponding decision variables need to be optimised.

B. Swarm Intelligent Optimisation

Within stochastic programming framework, it is important to have an optimiser that can provide reasonable optimal results considering trade off between searching speed and the complexity of the problem (which grows as the number of simulated paths or the size of the decision tree increases). For this reason, we consider swarm intelligent optimisers,

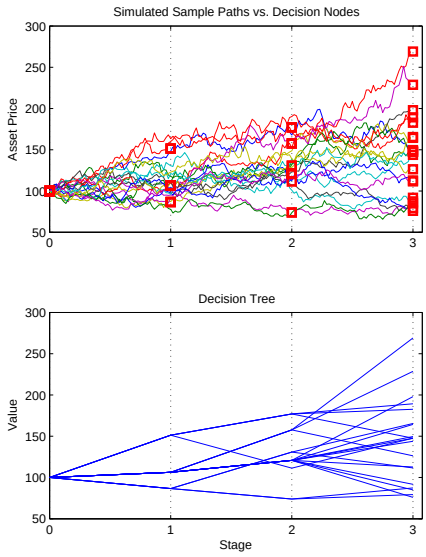


Fig. 3. Constructing Decision Tree

especially the Bacterial Foraging Optimsation (BFO) and the Particle Swarm Optimisation (PSO) algorithm due to their potential for solving high-dimensional problems [3].

Randomly distribute initial values for $\theta^i, i = 1, 2, ..., S$ across the optimization domain. Compute the initial cost function value for each bacterium i as J^i, and the initial total cost with swarming effect as J^i_{sw}.

for *Iteration* k **do**
  for *Chemotaxis loop* **do**
    for *Bacterium* i **do**
      Tumble: Generate a random vector $\phi \in \mathcal{R}^D$ as a unit length random direction
      Move: Let $\theta^{new} = \theta^i + c\phi$ and compute corresponding J^{new}. Let $J^{new}_{sw} = J^{new} + J_{cc}(\theta^{new}, \theta)$
      Swim: Let m=0
      while $m < N_s$ **do**
        let m=m+1
        if $J^{new}_{sw} < J^i_{sw}$ **then**
          Let $\theta^i = \theta^{new}$, compute corresponding J^i and J^i_{sw}
          Let $\theta^{new} = \theta^i + c\phi$ and compute corresponding $J(\theta^{new})$. Let $J^{new}_{sw} = J^{new} + J_{cc}(\theta^{new}, \theta)$
        else
          let $m = N_s$
        end
      end
    end
  end
  Reproduction: Sort bacteria in order of ascending cost J_{sw}The $S_r = S/2$ bacteria with the highest J_{sw} value die and other S_r bacteria with the best value split; Update value of J and J_{sw} accordingly.
  Elimination-dispersal: Eliminate and disperse the bacteria to random locations on the optimization domain with probability p_{ed}. Update corresponding J and J_{sw}.
end

Algorithm 2: BFO algorithm

1) The BFO algorithm: The *E.coli* bacteria present in our intestines undertake a foraging strategy, the activity of which inspired researchers to use it as optimization process. Passino introduced the canonical BFO algorithm in [15]. The objective is to find the minimum of $J(\theta), \theta \in \mathcal{R}^D$, where we do not have the gradient information $\nabla J(\theta)$. The bacterium will try to move towards increasing concentrations of nutrients (i.e. find lower values of J), search for ways out of neutral media (where $J = 0$) and avoid noxious substances (away from positions where $J > 0$). It implements a type of biased random walk. Bacteria can also engage in a form of chemically-mediated 'social communication' during their search process. The ways representing swarming effect can be different, Passino suggests a cell-cell attractant/repellant function. Let J^i denote the actual cost (or the nutrient surface) at the position of the ith bacterium θ^i. A bacterium that has uncovered good sources of nutrients during its search can release a chemical signal which attracts other bacteria to converge (or swarm) to its current location. This process is mediated by the release of a 'repellent signal' to ensure that the bacteria do not get too close to each other. Including this mechanism in our optimization algorithm, the problem becomes the minimization of $J^i_{sw} = J^i + J_{cc}(\theta_i, \theta)$, which represents the time-varying total cost value for bacterium i. The mathematical swarming (cell-cell signalling) function can be represented by:

$$J_{cc}(\theta^i, \theta) = -M(\sum_{k=1}^{S} e^{-W_a \|\theta^i - \theta^k)\|^2} - \sum_{k=1}^{S} e^{-W_r \|\theta^i - \theta^k\|^2})$$

where $\|.\|$ is the Euclidean norm, W_a and W_r are measures of the width of the attractant and repellent signals respectively, M measures the magnitude of the cell-cell signalling effect. We use $M = 0.01, W_a = 0.4, W_r = 10$ in our experiment.

During the lifetime of *E.coli* bacteria, they undergo different stages such as chemotaxis, reproduction and elimination-dispersal. The chemotactic step provides a basis for local search, the reproduction step speeds the convergence, and the elimination-dispersal step prevent the local optimum trapping effectively.

a) Chemotaxis: Chemotaxis is the tendency of a bacterium to move toward distant sources of nutrients. In this process, the bacterium alternates between tumbling (changing direction) and swimming behaviors. Here, a *tumble* is represented by a unit walk with random direction $\phi \in \mathcal{R}^D$, a *swim* is indicated as movement in the same direction as the previous tumble. After one step move, the new position of the ith bacterium can be represented as $\theta^{new} = \theta^i + c\phi$, where $\theta^i \in \mathcal{R}^D$ indicates the position of the ith bacterium across the optimization domain. c is the chemotactic step size taken in the direction of ϕ. In this paper, we consider an adapted step size c to control the convergence speed, i.e., c starts from $\frac{Range\ of\ search\ domain}{100}$, and shrinks after each reproduction step.

b) Reproduction: After N_c chemotactic steps, a reproduction step is taken. In reproduction, the least healthy bacteria (the health measured by J_{sw}) die and the other S_r

healthiest bacteria each split into two bacteria, which are then placed in the same location.

c) Elimination - dispersal: In elimination-dispersal, individual bacterium is stochastically selected for elimination from the population and is replaced by a new bacterium located at a random new location within the optimization domain, according to a preset probability p_{ed}.

In this paper, we use a modified version of the BFO algorithm originally in [15]. With improved computing speed, the algorithm is illustrated in Algorithm 2.

2) The PSO algorithm: PSO is a population based stochastic optimization technique developed by James Kennedy and Russell Eberhart in 1995 [11]. As described by Jennedy and Eberhart, 'Particle swarm algorithm imitates human (or insects) social behavior. Individuals interact with one another while learning from their own experiences, and gradually the population members move into better regions of the problem space'. In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods [18].

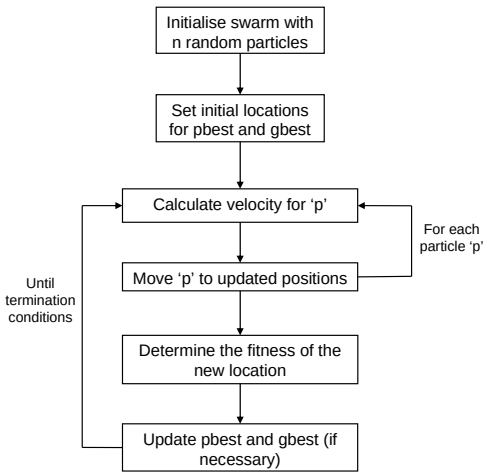


Fig. 4. Flowchart of the PSO algorithm

In PSO, the potential solutions (particles), fly through the problem space by following the current optimum particles. Each particle keeps track of its coordinates in the problem space corresponding to the best solution (fitness) it has achieved so far. This value is called pbest. Another 'best' value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. When a particle takes all the population as its topological neighbors, this best value is a global best and is called gbest. The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its pbest and gbest locations. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and gbest locations. The flowchart of the canonical PSO algorithm used in this paper is illustrated below, further details can be found in [2].

TABLE I

CAPARISON OF GA, BFO AND PSO ALGORITHMS

GA	BFO	PSO
Fitness function	Nutrient concentration function	Objective function
Selection	Bacterial reproduction	Reproduction
Crossover	Bacterial splitting	No Crossover
Mutation	Elimination & dispersal	No Mutation

3) The Hybrid algorithm: The hybrid optimiser using the Bacterial Foraging Optimisation algorithm for initialisation and the Sequential Quadratic Programming (SQP) method for local search. The SQP method closely mimic Newton's method for constrained optimisation, an overview of the method can be found in [6].

4) Comparative study with GA: The canonical BFO or PSO algorithm shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. To understand the relative performance characteristics of BFO and PSO, we compare them with the canonical GA.

These algorithms are all population-based search algorithms. As shown from Table I, the nutrient concentration function, the fitness function and the objective function used in GA, BFO & PSO respectively are the types of landscape. In BFO, bacteria in the most favorable environments gain a selective advantage for reproduction, which is similar to the Selection process in GA, and the Reproduction process in PSO. In BFO, the bacteria with higher fitness (lower cost) split into two children, which are at the same concentration, whereas in GA, with crossover they generally end up around their parents on the fitness landscape. In BFO, elimination-dispersal results in physical dispersion in a geographical area, and mutation in GA results genotypical changes, but they both can help jumping out of the local optimum trap during the search.

IV. RESULTS ANALYSIS

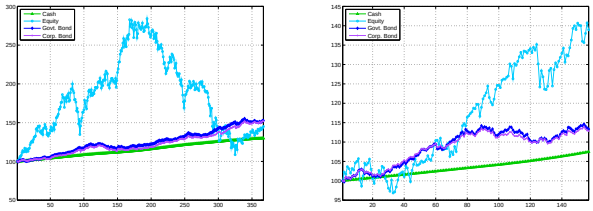
The data set we use is historical weekly data from 03/01/1997 to 25/12/2006, including 3M EURIBOR TR index (representative of the 'safe' asset) and the DJ EuroStoxx 50 index (denoting the 'risky' equity asset). For the 4-asset case, we consider 2 additional asset product, represented by a government bond index (JP Morgan EMU Govt Bond Index) and a corporate bond index (msci emu credit IG). The data sample is divided into 7-year training period (from 03/01/1997 to 29/12/2003, total of 366 observations) and a 3-year validation period from (05/01/2004 to 25/12/2006, total of 156 observations). Given initial wealth of 100, the growth of it holding individual asset during the training and validation period are illustrated in Figure 5(a) and Figure 5(b) respectively. We compare the results of different optimisers during the trainging period, and the resulting real performance of wealth are tested during the validation period. The experiments are implemented in Matlab 7.6, running on

TABLE II

OPTIMISER SPECIFIC PARAMETERS

BFO Parameters	
N^o. of chemotactic steps	10
N^o. of swimming steps	4
N^o. of of bacteria for reproduction/splitting	15
Prob. each bacterium will be e-d.	0.5
PSO Parameters	
Cognitive acceleration	2
Social acceleration	2
Initial inertia weight	0.9
Final inertia weight	0.3
Max. velocity	0.5
GA Parameters	
Crossover rate	0.7
Mutation rate	0.1
Selection rate	0.9

iMac, Mac OS X 10.4.11, with 2.4 GHz Intel Core 2 Duo, and 1 GB 667 MHz DDR2 SDRAM.



(a) Training Period (b) Validation Period

Fig. 5. Performance of the Indices

A. Results of Training Period

In the following, we compare different optimisation results of BFO, PSO and GA. In order to provide a benchmark for the results obtained by these optimisers, we use the Sequential Quadratic Programming (SQP) method, and the Matlab optimising function *fmincon* was used to perform this task. Based on our preliminary experimental results, we also consider a hybrid optimiser, which employs BFO for the initialisation process of the decision variables, and continue with SQP for the local search. We used the three-period (four-stage) SP model with following parameters: assuming planning period T of 3(yrs), with rebalancing decisions made end of each year. In the scenario generation step, 1000 sample paths are generated, using $[1, 20, 5]$ branching factors associated with the sequential clustering procedure to construct decision tree. This results a total number of 121 (1+20+20*5) decision variables (single asset weights) to be optimised.

1) Parameter setting: To make a fair comparison between BFO, PSO and GA, we use the same population size S of 20, and the same number of iterations (generations) of 200. The specific parameters used in each optimiser are shown in Table II. The value are chosen according to preliminary trial and errors, to make balance between search speed and accuracy.

2) Optimisation Results of 2-asset case: BFO, PSO and GA are originally designed for unconstrained problems. To deal with the weight constraint, we use a bound constraint of [0, 1] to the weight of Asset 1 at time t and scenario s ($\theta^s_{1,t}$), the weight of Asset 2 is (1-$\theta^s_{1,t}$). Hence for the 2-asset case, we have total of 121 weight variables across multiple time period and scenarios to be optimised. The bound constraint would be satisfied during the search process of each optimiser. We tested 30 independent runs of the different optimisers. The results of objective value and computing time are reported in Table III.The best results over 30 runs are reported in the second row of Table III. The best results averaged over 30 runs are reported in the third row. The standard deviation of the best results over 30 runs are reported in the fourth row.

TABLE III

RESULTS OF DIFFERENT OPTIMISERS (2-ASSET CASE, *30*-RUN RESULTS)

Obj	SQP	BFO	PSO	GA	Hybrid
Best	-45.36	-41.88	-41.21	-35.54	-45.41
Mean	-45.32	-41.22	-36.59	-34.44	-45.36
S.D.	0.118	0.592	4.34	1.17	0.109
Time(s)	82.1	102.5	80.2	78.7	75.4

(The objective value shown in the table is the negative utility value mentioned in Section II, hence the lower value, the better.)

From the result, we can find that although BFO does not outperform SQP, it provides better accuracy (as seen from the best objective value) and stability (as indicated by the standard deviation of the objective value) than PSO and GA. The result of the hybrid optimiser at the last column provides better objective value than SQP, without loosing much of the computing speed. In the following, we extend our experiments to the 4 asset case. PSO has quicker computing speed, However, for our current experiment, we are more concerned of the accuracy and stability of the solution to the SP model.

3) Optimisation Results of 4-asset case: To deal with the weight constraint for the 4-asset case, we generate the weight of Asset 1,2,3 at time t and scenario s first, denoted as $\theta^s_{i,t}, i = 1, 2, 3$, the weight of Asset 4 is $(1 - \sum_{i=1,2,3} \theta^s_{i,t})$. Hence for the 4-asset case, we have total of 363 (121*3) weight variables across multiple time period and scenarios to be optimised. The process is running recursively until the weight constraint is satisfied (i.e., $\sum_{i=1,2,3,4} \theta^s_{i,t} = 1$).

The results of objective value and computing time are reported in Table IV. From the result, we can find that similarly to the 2-asset case, BFO does not outperform SQP, but provides better accuracy and stability than PSO and GA. However, as the problem size grows, the searching ability of the two swarm intelligent optimisers, BFO and PSO improves relative to SQP. This would be important for future applications to a harder asset allocation problem. Also, we can find that the hybrid optimiser outperforms the others from both accuracy and stability aspects. The evolution of the objective value vs. the iteration number during the searching process (of a single run) is illustrated

in Figure 6. It is apparent that PSO has quicker convergence, but the accuracy of the result is sacrificed compared to BFO. Table V reports the best run result of the asset allocation problem. For the initial period, there is only one decision node, and corresponding asset weight results are provided. For the subsequent periods, we provide average asset weights with standard deviation across various scenarios at each period. We can find that the resulting decisions of BFO at intermediary decision stage across various scenarios tend to be stable (measured through the S.D. of asset weights). The expected utility level (- objective value) is smaller for BFO compared to SQP, but the probability of wealth over target value at the terminal stage is higher.

TABLE IV

RESULTS OF DIFFERENT OPTIMISERS (4-ASSET CASE, *30*-RUN RESULTS)

Obj	SQP	BFO	PSO	GA	Hybrid
Best	-48.82	-48.12	-47.84	-44.71	-49.32
Mean	-48.73	-47.98	-46.77	-44.20	-49.12
S.D.	0.149	0.130	0.928	0.455	0.113
Time(s)	133.7	119.3	95.8	198.2	113.6

TABLE V

ASSET ALLOCATION RESULTS OF 4-ASSET CASE (BEST RUN RESULTS)

		SQP	BFO	PSO	GA	Hybrid
1^{st} Period Decision						
Asset weights	1	0.00%	0.01%	0.00%	7.45%	0.00%
	2	6.84%	8.97%	8.20%	11.76%	10.90%
	3	74.59%	21.55%	0.00%	33.73%	73.89%
	4	18.56%	69.48%	91.80%	47.06%	15.21%
2^{nd} Period Decision						
Ave. (asset weights)	1	4.59%	9.61%	0.00%	27.78%	1.29%
	2	20.44%	11.34%	6.91%	17.98%	19.39%
	3	41.04%	13.23%	4.45%	29.32%	37.02%
	4	33.93%	65.81%	88.64%	24.91%	42.30%
S.D. (asset weights)	1	0.125	0.102	0.010	0.193	0.030
	2	0.132	0.072	0.126	0.141	0.098
	3	0.167	0.110	0.161	0.169	0.211
	4	0.161	0.153	0.189	0.165	0.199
3^{rd} Period Decision						
Ave. (asset weights)	1	35.97%	13.81%	6.04%	24.98%	24.48%
	2	21.27%	13.57%	6.04%	22.77%	23.11%
	3	22.79%	12.61%	6.25%	31.36%	24.17%
	4	19.98%	60.02%	81.67%	20.89%	28.25%
S.D. (asset weights)	1	0.260	0.115	0.213	0.194	0.221
	2	0.191	0.115	0.190	0.181	0.234
	3	0.183	0.102	0.218	0.231	0.173
	4	0.162	0.181	0.333	0.180	0.218
Terminal Stage						
Wealth>Target		86.3%	89.5%	86.4%	80.9%	86.1%

(Asset 1: Cash, Asset2: Equity, Asset3: Govt. bond, Asset4: Corp. bond)

B. Results of Validation Period

For the Stochastic Programming model, the emphasis is on obtaining a good first-stage solution rather than obtaining an entire accurate policy [10]. To test the asset allocation results, we re-run the SP model rolling the horizon forward one year during the validation period. The allocation decisions are made using the initial stage solution of the SP model for each rollover year. Based on the comparative results of different optimisers, we run our results based on the *hybrid* method. Basically, we use initial weights generated by BFO as input to SQP, the results are more stable than running SQP with random starting value, and the computing speed is also increased.

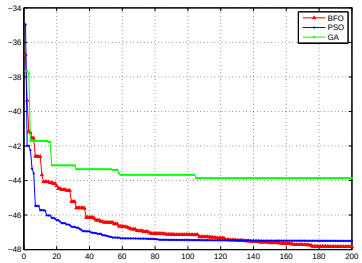
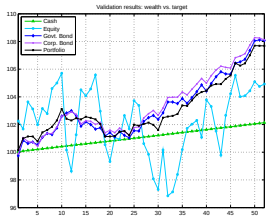


Fig. 6. Search results comparison

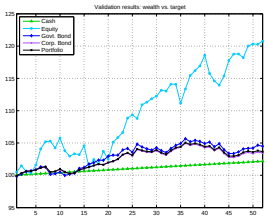
TABLE VI

ASSET ALLOCATION AVERAGE RESULTS WITH ROLLOVER HORIZON

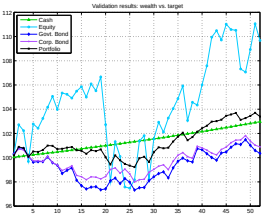
Rollover	Asset 1	Asset 2	Asset 3	Asset 4
1	0.00%	16.62%	80.83%	2.55%
2	0.00%	1.13%	3.32%	95.55%
3	39.66%	19.41%	24.99%	16.94%



(a) 2004 (annual return=7.69%)



(b) 2005 (annual return=3.65%)



(c) 2006 (annual return=3.39%)

Fig. 7. Wealth Performance with Rollover Effect

Table VI provides asset allocation results for each rollover year. Fig. 7 illustrate wealth performance of the portfolio and the four individual asset given initial wealth of 100 at the beginning of each rollover year. The growth of the wealth (initially of 100) over the entire validation period (3 years) is displayed in Figure 8. Overall, a total return of 15.41% could be achieved during the entire 3-year validation period, by rebalancing the portfolio at the beginning of each year based on the results from rolling the SP model forward. For the 1^{st} rollover year (2004), the portfolio outperforms the target Cash product 100%, even though we can find through Figure 7(a) that Corp. Bond outperforms Govt. Bond

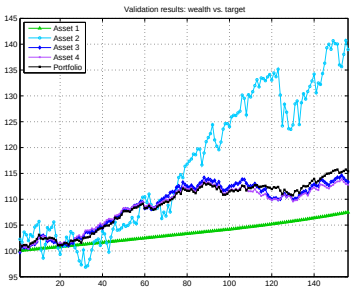


Fig. 8. Results of Wealth Performance during the Validation Period (4 assets), total return $= 15.41\%$

during 2004, majority of investment goes to Govt. Bond as recommended by the SP model, this is because our target is set to be Cash product, if it is set to Govt. Bond or other liability index, the solution would be different. In year 2005, the portfolio outperforms target Cash by 98.08%, it does not benefit much from the booming of the Equity market, since most investment goes to Corp. Bond, again, by set different target, the allocation decisions would be different. For year 2006, the portfolio outperforms Cash target by 51.92%, but it achieved higher return than Cash during the period as seen from Figure 7(c). Though the portfolio return is lower than investing solely in equity, it avoids the significant downside risk of equity during the middle of the period. The resulting wealth performance would be improved if we increase the frequency of rebalancing times, and other parameters of the SP model such as the risk aversion measure β, the number of simulated paths etc. would also affect the optimal results obtained, however, these are beyond the scope of this paper.

V. Conclusions and Future Research

In this paper, we solved a dynamic asset allocation problem through a three-period stochastic programming model. To improve the optimisation result of the model, we employ swarm intelligent optimisers, and the numerical results suggest the hybrid method which employ Bacterial Foraging Optimisation algorithm for initialisation of the decision variables and Sequential Quadratic Programming for the following search process provides better result, with improved accuracy, stability and computing speed than using BFO, PSO, or SQP alone. Although PSO does not achieve better objective value compared with BFO during the search process, it has the advantage in the time of convergence, hence, depending on the specific problem setting, a hybrid method based on PSO and SQP might also be applied.

Swarm intelligent algorithms such as BFO, PSO are originally designed for unconstrained problems. In this paper, we consider weight and short-selling constraints, and the problem are solved by transforming to bound-constraint problem. Our future work would be incorporating more realistic constraints into our model, such as the turnover constraints, regulatory constraints, etc. These would bring more challenging work to the development and applications of swarm intelligent optimisers into real world dynamic asset allocation problems. Also, we are interested to extend the current dynamic asset allocation problem with consideration of derivative products, where the objective function could be non-convex. Different forms of stochastic models could be built, depending on the type of uncertainty and the time when decisions must be taken. This links the various concepts to alternative fields of planning under uncertainty.

References

[1] Birge, J.R. and Louveaux, F.(1997): Financial planning and control, Introduction to Stochastic Programming. Spring-Verlag, New York.

[2] Brabazon, A. and O'Neill, M. (2006). *Biologically-inspired Algorithms for Financial Modelling*, Berlin: Springer.

[3] Dang, J., Brabazon, A., O'Neil, M. and Edelman, D. (2008). Option model calibration using a bacterial foraging optimisation algorithm, Applications of Evolutionary Computing of Lecture Notes in Computer Science, 4974: 113-123, Springer.

[4] Dempster, M.A.H., Germano, M., Medova, E.A., Villaverde, M.: Global asset liability management. *British Acturial Journal*, 9(1), 137-216 (2003).

[5] Dupaĉová, J., Hurt, J. and Ŝtêpán, J. *Applied Optimization 75: Stochastic Modeling in Economics and Finance*, Kluwer Academic Publishers (2002).

[6] Gill, P.E., Murray, W. and Wright, M.H., Practical Optimization, London, Academic Press, 1981.

[7] Glassedrman, P. (2004). *Monte Carlo Methods in Financial Engineering*, Chap 3, Springer-Verlag, New York.

[8] Hochreiter, R. Algorithmic aspects of scenario-based multistage decision process optimization. Volume 5783 of Springer Lecture Notes in Artificial Intelligence: 365 - 376. 2009.

[9] Hibiki, N. "Multi-period stochastic optimization models for dynamic asset allocation", *Journal of Banking and Finance*, vol.30, pp.365-390, 2006.

[10] Infanger, G. Dynamic asset allocation strategies using a stochastic dynamic programming approach, In: Zenios, S.A. and Ziemba, W.T. (eds.) *Handbook of Asset and Liability Management*, Vol.1, 199-251 (2006).

[11] Kennedy, J. and Eberhart, R. C. 2001. *Swarm Intelligence*, Morgan Kaufmann.

[12] MacQueen, J.B. Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, 1:281-297 (1967)

[13] Markowitz, H. (1952). Portfolio selection. Journal of Finance, 1(7):77-91.

[14] Mulvey, J.M, Pauling, W.R. and Madey, R.E., Advantages of multiperiod portfolio models, Journal of Portfolio Management, 29, 35-45, 2003.

[15] Passino K-M (2002). Biomimicry of bacterial foraging for distributed optimization and control, Control Systems Magazine. IEEE Vol. 22, Iss. 3, pp.52-67.

[16] Ruszczynski, A. and Shapiro, A. (Eds.) Stocahstic Programmjng. Handbooks in Operations Research and Management Science, Vol.10, New York, 2003.

[17] Sandrini, F., Edelman, D. and Ryan, D. Estimating the True Embedded Risk Management Cost of Total Return Strategies, *Journal of Computational Finnace* (forthcoming).

[18] Shi, Y. and Eberhart, R.C. (1999) Empirical study of particle swarm optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol.3, pp.1945-1950, 1999.

[19] Wallace, S.W. and Ziemba, W.T. (eds.), Applications of Stochastic Programming, 2005.

[20] Ziemba, W.T., The Stochastic Programming Approach to Asset, Liability, and Wealth Management, The research foundations of AIMR, 2003.