Expert Systems with Applications 36 (2009) 3824–3828

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Constructing investment strategy portfolios by combination genetic algorithms

Jiah-Shing Chen [a], Jia-Li Hou [b,*], Shih-Min Wu [a], Ya-Wen Chang-Chien [a]

[a] Department of Information Management, National Central University, No. 300, Jhongda Road, Jhongli City, Taoyuan County 32001, Taiwan, ROC
[b] Department of Information Management, National Dong Hwa University, No. 1, Section 2, Da Hseuh Road, Shoufeng, Hualien, Taiwan, ROC

A R T I C L E I N F O

Keywords:
Genetic algorithms (GA)
Portfolio
Capital allocation
Combination genetic algorithm
Investment strategy portfolio

A B S T R A C T

The classical portfolio problem is a problem of distributing capital to a set of securities. By generalizing the set of securities to a set of investment strategies (or security-rule pairs), this study proposes an investment strategy portfolio problem, which becomes a problem of distributing capital to a set of investment strategies. Since the investment strategy portfolio problem can be formulated as a combination optimization problem, a new combination genetic algorithm is proposed for solving the new investment strategy portfolio problem. Experimental results show that the idea of investment strategy portfolios is feasible and the combination genetic algorithm is effective on the investment strategy portfolio problem.

1. Introduction

The classical portfolio problem is a problem of distributing capital to a set of securities (Gondzio & Grothey, 2007; Ince & Trafalis, 2006; Markowitz & Arnott, 1952; Wu & Chang, 2007). By generalizing the set of securities to a set of investment strategies (or security-rule pairs), this study proposes an investment strategy portfolio problem, which becomes a problem of distributing capital to a set of investment strategies. The classical portfolio problem can be viewed a special case of the new investment strategy portfolio problem with buy-and-hold as the only trading rule.

Both the investment strategy portfolio problem and the classical portfolio problem can be formulated as combination optimization problems. Therefore, a new combination genetic algorithm (CGA) is proposed for solving the combination optimization problem in general, and the new investment strategy portfolio problem in particular.

Statistical test result indicates that the performance of our CGA is significantly better than that of uniform allocation. Experimental results show that the idea of investment strategy portfolios is feasible and the combination genetic algorithm is effective on the investment strategy portfolio problem.

The rest of this paper is organized as follows. Section 2 reviews the classical portfolio problem and genetic algorithms. Section 3 describes the investment strategy portfolio problem and our solution method, the combination genetic algorithm. Section 4 presents the results of our experiments. Section 5 gives the conclusions and future directions.

2. Background

2.1. Classical portfolio optimization problem

The classical portfolio optimization problem can be formulated as follows:

$$\begin{aligned} &\max && \alpha \sum_i x_i r_i - (1-\alpha)\sum_{ij} x_i x_j \sigma_{ij} \\ &\text{subject to} && \sum_{i=0}^{n} x_i = 1, 0 \leqslant x_i \leqslant 1, x_i \in R, \end{aligned} \tag{1}$$

where α is a real constant between 0 & 1, x_i is the proportion (between 0 and 1) of capital invested in instrument i, r_i is the expected return rate of instrument i and σ_{ij}^2 is the covariance of r_i and r_j. The objective is to find the optimal portfolio with maximum return and/or minimum risk.

Traditional approach to this portfolio problem is Markowitz's mean–variance analysis which uses the Lagrange multiplier method to find the optimal static portfolio (Kim & Markowitz, 1989; Markowitz & Arnott, 1952).

2.2. Genetic algorithms on portfolio optimization problem

Genetic algorithms (GA) were proposed by Holland in 1975 from Darwin's theory of evolution, i.e., survival of the fittest (Darwin, 1859; Holland, 1975). Genetic algorithms uses an evolutionary process resulting in a fittest solution to solve a problem. The evolutionary process consists of several genetic operators:

* Corresponding author. Tel.: +886 932 228466; fax: +886 3 863100.
E-mail addresses: jschen@ncu.edu.tw (J.-S. Chen), alexhou@mail.ndhu.edu.tw (J.-L. Hou), 93423012@cc.ncu.edu.tw (S.-M. Wu), 92443001@cc.ncu.edu.tw (Y.-W. Chang-Chien).

0957-4174/$ - see front matter © 2008 Elsevier Ltd All rights reserved.
doi:10.1016/j.eswa.2008.02.019

selection, crossover and mutation (Bäck, Fogel, & Michalewicz, 2000a, 2000b; De Jong, 2002; Goldberg, 1989; Mitchell, 1996; Srinivas & Patnaik, 1994).

Genetic algorithms are computationally simple and powerful. Genetic algorithms are very good tool for optimization problems since they make no restrictive assumptions about the solution space.

To solve a problem with genetic algorithms, an encoding mechanism must first be designed to represent each solution as a chromosome, e.g., a binary string. A fitness function is also required to measure the goodness of a chromosome. Genetic algorithms search the solution space using a population which is simply a set of chromosomes. During each generation, the three genetic operators: selection, crossover and mutation, are applied to the population several times to form a new population. Selection picks 2 chromosomes according to their fitness: a fitter chromosome has a higher probability of being selected. Crossover recombines the 2 selected chromosomes to form new offspring with a crossover rate. Mutation randomly alters each position in each offspring with a small mutation rate. New population is then generated by replacing some chromosomes with new offspring. This process is repeated until some termination condition, e.g., the number of generations, is reached. Fig. 1 shows the pseudo code of the basic genetic algorithm. When the number of genetic applications (k) is half the population size ($n/2$), the GA is called generational GA; when $k < n/2$, it is called steady-state GA.

The advantage of GAs is in their parallelism. GA searches a solution space using a population of individuals so that they are less likely to get stuck in local optimums. This is achieved with a cost, i.e., the computational time. GAs can be slower than other methods. However, the longer run time of GAs can be shortened by terminating the evolution earlier to get a satisfactory solution.

Genetic algorithms have been applied to various domains over the years (Beasley, 2000; Oh, Kim, & Min, 2005; Oh, Kim, Min, & Lee, 2006). Financial applications of genetic algorithms are starting to show promising results. Bauer used genetic algorithms to generate trading rules which are Boolean expressions with 3 of the 10 allowed time series (Bauer, 1994). Colin applied genetic algorithms to find the lengths in the moving average crossover strategy (Colin, 1994). Deboeck studied methods of using genetic algorithms to train a neural network trading system (Deboeck, 1994).

The easiest way to use GAs on the portfolio problem is to encode each weight x_i as a non-negative integer or floating number. Standard genetic operators can then be applied as usual. To enforce the $\sum_{i=0}^{n} x_i = 1$ constraint, normalization of each x_i by dividing their sum $\sum_{i=0}^{n} x_i$ is usually required (Xia, Liu, Wang, & Lai, 2000).

One problem with this encoding method is that many chromosomes decode into the same portfolio. This multiplies the GA's search space and makes GA less efficient in finding the optimal portfolio. Another problem with normalization is that similar chromosomes may decode into very different portfolios which makes it more difficult for GA to produce better chromosomes from good chromosomes.

3. Investment strategy portfolio problem and combination genetic algorithms

3.1. Investment strategy portfolio problem

The classical portfolio problem is a problem of distributing capital to a set of securities. By generalizing the set of securities to a set of investment strategies (or security-rule pairs), this study proposes an investment strategy portfolio problem, which becomes a problem of distributing capital to a set of investment strategies.

An investment strategy is a security-rule pair. A (trading) rule consists of a buy condition and a sell condition which are used to, respectively, determine the buying time and selling time for the target security. The buy/sell conditions can be based on various factors.

The classical portfolio problem can then be viewed a special case of the new investment strategy portfolio problem with buy-and-hold as the only trading rule. Other variants of the investment strategy portfolio problem include multiple securities with a single trading rule, a single security with multiple trading rules, and multiple securities with multiple trading rules.

3.2. Combination genetic algorithms

The real number model of portfolio problem in (1) can be approximated by the following integer model with a large m to achieve the desired precision.

$$\begin{aligned} &\max && \alpha \sum_{i} \frac{w_i}{m} \cdot r_i - (1-\alpha) \sum_{ij} \frac{w_i}{m} \cdot \frac{w_j}{m} \cdot \sigma_{ij} \\ &\text{subject to} && \sum_{i=0}^{n} w_i = m,\ 0 \leqslant w_i \leqslant m,\ w_i \in Z, \end{aligned} \tag{2}$$

where m is a large positive integer, w_i/m approximates the x_i in (1) and the constraints become $\sum_{i=0}^{n} w_i = m$, $0 \leqslant w_i \leqslant m$, & $w_i \in Z$. This turns the portfolio optimization problem into the combination optimization problem and reduces the search space to non-negative integers such that $\sum_{i=0}^{n} w_i = m$, whose size is C_m^{n+m}.

There are four common types of combinatorial problems: arrangement, permutation, combination, and combination with repetition. The sizes of their solution spaces are listed below.

Arrangement: There are n^r ways of ordering r of n distinct objects with repetitions.
Permutation: There are $P(n,r) = n!/(n-r)!$ ways of ordering r of n distinct objects without repetitions.
Combination: There are $C(n,r) = n!/r!(n-r)!$ ways of selecting r of n distinct objects without repetitions.

```
procedure BasicGA
  Initialization: Generate a random population of n chromosomes
  while (termination condition is not satisfied)
    Evaluation: Evaluate the fitness of each chromosome in the population
    loop (do genetic applications k times for each generation)
      Selection: Select 2 chromosomes according to their fitness
      Crossover: Cross over selected chromosomes to form new offspring with a crossover rate
      Mutation: Mutate each position in each new offspring with a mutation rate
    endloop
    Replacement: Replace chromosomes in parent population with new offspring
  endwhile
  Report the best solution (chromosome) found
endprocedure
```

Fig. 1. The basic genetic algorithm.