Portfolio optimization with heuristic algorithms

Section 1

Genetic algorithms in pills

Genetic algorithms - introduction

GA are a tool for searching an optimal solution in very large, even irregular spaces. They are inspired by biological evolution, recombining parts of a solution to obtain a better one. There is no guarantee to reach the global optimum, but the solution is not worse at each step.
The basic idea is to generate a population of admissible solutions each described by an array named "chromosome", evaluate each of them via a fitness function, and generate a new population taking into consideration the fitness of the solutions. Usually the best solution from the previous population is included in the new one (*elitism*).
When the best solution converges and do not improve any more the procedure ends.
The generation of a new population is performed mimicking some biological processes: during the crossover a subsequence of the array is exchanged between two chromosomes; the mutation, instead, is a change involving just one chromosome at time.

General steps

A GA generally follows these steps.

1. Generate an initial population of admissible solutions
2. Evaluate the fitness of each chromosome
3. If the convergence check is satisfied return the best solution and end the algorithm
4. If not, generate a new population repeating the following steps
    1. select two chromosomes from the population: best fitting solutions must be chosen with higher probability
    2. perform a crossover with a given probability
    3. perform a mutation with a given probability
    4. if admissible, place the solution in the new population
5. Go back to the evaluation step

Design choices

Many variations can be obtained according to:

- how the fitness function is defined
- how chromosomes are encoded and created
- how parents are selected
- how crossover and mutation are defined

Examples for binary encoding

The binary encoding represents the solution as a string of binary values.

- A crossover on this kind of chromosome is the so called "single point crossover": a crossover point is selected, then the first part of the child comes from the first part of the first parent, while the second part comes from the second part of the second parent.
- In a two point crossover the first and the last substring of the child come from one parent, the mid part from the other.
- In a uniform crossover the elements of the child are randomly chosen from the first or the second parent.
- In an arithmetic crossover the elements of the child are the result of a binary operator applied on the corresponding couple of elements coming from the two parents.

An example of mutation is the inversion of a bit with a given probability.

Other types of encoding

The permutation encoding represents the solution as a string of numbers which identifies a permutation of elements: crossover and mutation must be designed so that the resulting solution is still admissible.
The direct value encoding is used for more complex problems: in this case the array contains values of any type, depending on the problem. The increased flexibility must be compared with a higher level of complexity in mutation and cross over.

Selection methods

There are several possible methods to select the best individuals to generate the new population:

- Roulette wheel: the probability to be selected is directly proportional to the fitness. If the fitness varies a lot among individuals this method selects too few individuals
- Rank selection: the individuals are ranked and a fixed probability is associated to each rank. This can lead to slow convergence
- Tournament selection: k individuals are selected randomly. Among these, the best is chosen with probability p, the second best with probability $p(1-p)$ and so on.

Tips

- Crossover rate should be high (60%-90%)
- Mutation rate should be low (0.5% - 1%)
- Population size provides limited advantage: usually a good choice is between 30 and 100

Section 2

Application to index tracking

The problem

We aim to reproduce a stock index without purchasing all the components of the index itself. Transaction costs are modelled, moreover, constraints on the maximum level of transaction costs as well as the maximum number of stocks selected are included.
We adopt a historical approach: given an index composed by N stocks which we observed for $t \in 0, 1, ..., T$ we select $K < N$ stocks that best replicated the index over the observation period.

Notation

- N, K Number of stocks investable and to be selected
- ϵ_i, δ_i minimum and maximum proportion of stock i, if selected
- X_i units of stock i held in the portfolio
- V_{it}, I_t, C value of the i-th stock and of the index at time t; value of the portfolio at T
- $F_i(\zeta, \theta, t)$ transaction costs for the i-th stock at time t in moving from holding ζ units to θ
- γ limit of the proportion of C that can be spent in transaction costs

Decision variables:

- y_i proportion of C invested in asset i; $y_i = V_{iT} x_i / C$ where x_i is the desired number of units for stock i
- z_i 1 if the stock i is selected, 0 otherwise

Constraints

1. $\sum_i z_i = K$
2. $y_0 = \sum_i F_i(X_i, x_i, T)/C$
3. $y_0 \leq \gamma$
4. $\sum_{i=0}^{N} y_i = 1$
5. $\epsilon_i z_i \leq y_i \leq \delta_i z_i \ \forall i \in 1, N$
6. $z_i \in \{0, 1\}$

Objective function

Define the returns of the index and of the portfolio respectively as:

$$R_t = \log(I_t / I_{t-1})$$

$$r_t = \log\left(\sum_{i=1}^{N} (V_{it} y_i / V_{iT}) / \sum_{i=1}^{N} (V_{it-1} y_i) / V_{iT}\right)$$

The tracking error and the excess return read:

$$E = \frac{1}{T}\sqrt{\sum_t (r_t - R_t)^2}$$

$$r^* = \sum_t (r_t - R_t)/T$$

Finally, given $\lambda \in [0, 1]$, the objective function to be minimized is:

$$\lambda E - (1 - \lambda) r^*$$

Representation

Define the "free" portfolio as $1 - \sum_{j \in Q} \epsilon_j$ where Q is the set of selected stocks. A vector of real number s_i, $s_i \in [0, 1]$ is employed to represent the portion of "free" portfolio attributed to the i-th stock, as usual s_0 is the portion related to transaction costs. To enforce the constraints 3-5 go trough the following procedure:

- set $\epsilon_0 = 0$, $\delta_0 = \gamma$, $Q = Q \cup 0$, $R = \emptyset$
- set $y_i = \epsilon_i + s_i(1 - \sum_{j \in Q} \epsilon_j) / \sum_{j \in Q} s_j \quad \forall i \in Q$
- while $\exists i \in Q \setminus R$ s.t. $y_i > \delta_i$:
  - $R = R \cup \{i \in Q \setminus R \text{ s.t. } y_i > \delta_i\}$
  - set $y_i = \epsilon_i + s_i(1 - \sum_{j \in Q \setminus R} \epsilon_j - \sum_{j \in R} \delta_j) / \sum_{j \in Q \setminus R} s_j \quad \forall i \in Q \setminus R$
  - $y_i = \delta_i \quad \forall i \in R$

New offspring

The procedure described above does not guarantee that the constraint 2 is satisfied. To achieve an admissible solution compute a "unfitness" value for each individual in the population as the absolute value of the difference between the two sides of the constraint.
When a new individual is generated one individual from the previous population must be killed. This is chosen accordingly with the following priority:

1. if the child has an unfitness lower than a given threshold, kill the individual with the worse fitness
2. the individual with the worse fitness among those with unfitness higher than the new child
3. the individual with the worse fitness among those with unfitness lower than the new child

If the child is worse than every else under both the points of view it is not added to the population. This procedure usually leads to slightly infeasible solutions, which can however be considered feasible with a good approximation.

Selection and mating

The selection follows a binary tournament, with two couples of potential parents randomly chosen: among each couple the individual with the best fitness is chosen. There is no need to considerate the unfitness. The child is built by uniform crossover

References

 Beasley, J.E., Meade, N., Chang, T.-J., An evolutionary heuristic for the index tracking problem, European Journal of Operational Research 148 (2003) 621-643