EvoPortfolio

Alfonso Guarino[a,*], Luca Grilli[b], Domenico Santoro[c], Francesco Messina[d], Rocco Zaccagnino[e]

[a]*Department of Humanities, University of Foggia, Via Arpi, 176, 71121 – Foggia (FG), Italy*
[b]*Department of Economics, Management and Territory, University of Foggia, Via da Zara, 11, 71121 – Foggia (FG), Italy*
[c]*Department of Economics and Finance, University of Bari, Largo Abbazia S. Scolastica, 70124 – Bari (BA), Italy*
[d]*Department of Statistical Sciences, University La Sapienza of Rome, Piazzale A.Moro 5, 00185 – Rome (RM), Italy*
[e]*Department of Computer Science, University of Salerno, Via Giovanni Paolo II, 84084 – Fisciano (SA), Italy*

Abstract

1. Scenario 2. Problema 3. Cosa si fa normalmente 4. Cosa proponiamo noi 5. La novelty 6. Cosa abbiamo trovato/che risultati abbiamo ottenuto

Keywords: Genetic algorithm, Portfolio optimization,

1. Introduction

1.1. Overview

Objective. In this paper, our goal is to

Motivation.

The proposed approach.

Key points. The main contributions of this paper are:

-
-
-

Paper's road map. This article is structured as follows: Section 2 offers basic notions to understand the rest of the paper. Section 4 presents the literature review, highlighting initiative and studies that show contact points with the presented paper. Section 5 concludes the paper with final remarks and an overview of the work done, and it traces the path for future works.

2. Preliminaries

Se la sezione troppo breve, valutare di includere all'inzio di Related work (come prima subsection) e rinominarlo come Literature Review

In this section,

*Corresponding author. Address: Department of Humanities, University of Foggia, Via Arpi, 176, 71121 – Foggia (FG), Italy

Email addresses: alfonso.guarino@unifg.it (Alfonso Guarino), luca.grilli@unifg.it (Luca Grilli), domenico.santoro@uniba.it (Domenico Santoro), francesco.messina1997@gmail.com (Francesco Messina), rzaccagnino@unisa.it (Rocco Zaccagnino)

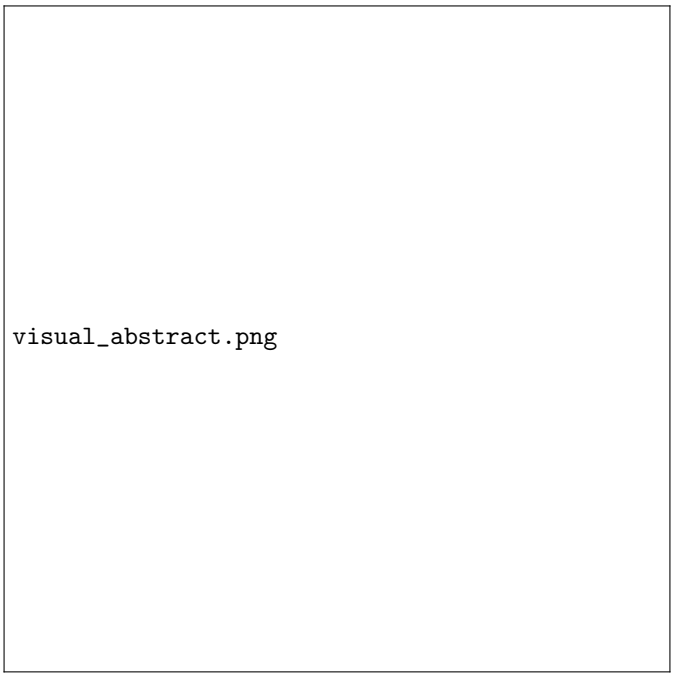

Figure 1: Visual abstract of our proposal.

2.1. Portfolio optimization problem

We define a *stock configuration* as a set $S = \{A_1, \ldots, A_N\}$ where A_i is a kind of stock, for $i = 1, \ldots, N$. Each kind of stock A_i has the following features: *(i) cost* of A_i (denoted with c_i), i.e., how much each unit of A_i cost, *(ii) yield* of A_i (denoted with y_i), i.e., how much each unit of A_i yields, and *(iii) risk* of A_i (denoted with r_i), i.e., how risky it is to invest in each unit of A_i.

Given a stock configuration, we define a *S-portfolio* as a vector $P_S = [P_1, \ldots, P_N]$ where P_i represents the part of P_S invested in A_i, for $i = 1, \ldots, N$. Specifically, each $P_i \geq 0$ is an integer indicating the quantity of stocks A_i purchased. Let $P_S = [P_1, \ldots, P_N]$ be a *S-portfolio*, we define:

- $C(P_S) = \sum_{i=1}^{N} P_i \times c_i$, where $C(P_S)$ is the *overall cost* of P_S,
- $Y(P_S) = \sum_{i=1}^{N} P_i \times y_i$, where $Y(P_S)$ is the *overall yield* of P_S,
- $R(P_S) = \sum_{i=1}^{N} P_i \times r_i$, where $R(P_S)$ is the *overall risk* of P_S.

The quality of P_S is measured in terms of $Y(P_S)$, i.e., as $Y(P_S)$ grows, the quality increases, and in terms of $R(P_S)$, i.e., as $R(P_S)$ grows, the quality decreases.

The *Portfolio optimization problem* can be formalized as follows: given a stock configuration S and an *initial budget* B, find the "best" S-portfolio, that is the S-portfolio P_S such that $C(P_S) \leq B$, which "maximize" $Y(P_S)$, and "minimize" $R(P_S)$.

- **Per l'ottimizzazione, possiamo avere dati riferiti a qualsiasi intervallo temporale: 1 solo valore per rischio e rendimento (quelli medi) oppure una serie di valori con cadenza settimanale, mensile, giornaliera, ...**
- **Nella realtà, quando comprano o vendono azioni, i gestori di portafogli pagano delle commissioni che cambiano in base alla grandezza del gestore. Nella teoria però, questi costi di transazione sono sempre nulli**

- **Possiamo usare le strategie degli agenti di FinRL per capire quali titoli dovrebbe essere sostituiti e da chi**
- **il rischio lo prendiamo dalla volatilità del titolo (un valore continuo)**
- **il rendimento è quello col logaritmo**
- **possiamo con una certa probabilità chiedere al FinRL un suggerimento su che vendere e che comprare in mezzo a quei titoli**
- **misurare la bontà del portafogli in una situazione "reale"**
- **utilizziamo il volume per aiutare la scelta di una azione rispetto ad un'altra negli operatori**

2.2. Multi-objective optimization

Usually, optimization problems involve multiple criteria, or objectives, that need to be considered. Such objectives can be conflicting, and no single solution, usually, simultaneously optimize all of them. When multiple objectives are considered, thus, a set of optimal solutions can be produced, instead of a single optimal solution, because no single solution could be optimal for multiple conflicting objectives. A solution y_1, which is better than a solution y_2 with respect to all the objectives, is preferable. In this case, we say that y_1 "dominates" y_2 (or y_2 is dominated by y_1). A non-dominated solution is an optimal solution. The Pareto-front [1, 2] is the set of all non-dominated solutions. Multi-objective evolutionary algorithms (MOEA) are a class of search methods that approximate the Pareto-front, i.e., instead of guaranteeing to find all non-dominated solutions, they aim at finding solutions that come as close as possible to the optimal ones. Once a set of such solutions is found, a higher-level decision-making strategy could be adopted to pick a single solution.

As we will see in **sezioni in avanti e spieghiamo**

Several MOEA have been proposed [1, 3, 4, 5, 6]. In this work, to define the Player agent we exploited a multi-objective genetic algorithm obtained by specialized variations of the NSGA-II strategy [5]. **qui dobbiamo parlare un po di NSGA-II e di come e stato applicato in letteratura. Rocco ha del materiale. E stato anche precedentemente applicato all'ottimizzazione dei portfolio come vedremo nella Section 4**

3. The algorithm

In this section, we present our algorithm that we call EvoPortfolio. The algorithm takes as input a voice line and produces a complete 4-voice harmonization of the input line.

3.1. Algorithm description

In this section we describe the details of the algorithm. We start by providing high-level motivations and an overall description and then proceed to explain the details.

3.1.1. Choices and motivations

To develop EvoPortfolio, first we have chosen a reference multi-objective evolutionary strategy, and then we have customized the strategy to work with specific stock market choices regarding the chromosome representation, the operators and the fitness function. Thus, the evolutionary process (stock market investment process) is obtained by running the chosen algorithm.

The strategy chosen as reference is NSGA-II ([**?**]). We made such a choice for several reasons. From a *practical* point of view these reasons are: *(1)* its popularity among practitioners and the vast available body of peer-reviewed material about it, *(2)* its proven efficiency on both benchmarks and real-word engineering problems, and *(3)* its clear algorithmic design and ease of implementation. NSGA-II has been already used ... **inserire citazioni**.

Pseudocode. We assume that the reader is already familiar with NSGA-II concepts which is the basis of EvoPortfolio. Further details can be found in [**?**]. Here we provide a brief description.

Algorithm 1 shows the pseudocode for EvoPortfolio. The algorithm takes as input a voice stock configuration S, the maximum budget B, the size of the population p_{size}, the crossover and mutation probabilities p_c, p_m and the maximum number of generations max_{gen}. It returns a set of solutions P, where each solution is a S-portfolio.

Algorithm 1: EvoPortfolio pseudocode

```
   Input  : S, B, p_size, p_c, p_m, max_gen.
   Output: P.
 1 P ← InitializePopulation(line, p_size);
 2 Evaluate(P);
 3 FastNonDominatedSort(P);
 4 for i = 1 to max_gen do
 5 |   Mate← SelectParentsByRank(P, B);
 6 |   EvoCrossover(Mate, p_c);
 7 |   EvoMutation(Mate, p_c);
 8 |   U ← Merge(P,Mate);
 9 |   F ← FastNonDominatedSort(U);
10 |   P ← ∅;
11 |   F_L ← ∅;
12 |   foreach F_j ∈ F do
13 |   |   CrowdingDistanceAssignment(F_j)
14 |   |   if size(P)+size(F_j) > p_size then
15 |   |   |   F_L ← F_j;
16 |   |   |   break;
17 |   |   else
18 |   |   |   P ← Merge(P, F_i);
19 |   if size(P) < p_size then
20 |   |   F_L ← SortByRankAndDistance(F_L);
21 |   |   P ← FillRemaining(F_L);
22 |   Evaluate(P)
23 return P;
```

The first step is that of creating the initial population and this is done in line 1, function InitializePopulation(). Section 3.4 provides the details of this step. Line 2, function Evaluate(), assigns to each element of the initial population its fitness values; this step is accomplished by applying the harmonic and melodic evaluation function that will be explained later in Sections 3.3-**??**. The third step, in line 3, FastNonDominatedSort(), orders the elements of the initial population with respect to the non-dominated relation. At this point the evolution of the population starts. In each generation, that is each cycle of the **for** loop, the following happens. In line 5 a new population *Mate* is obtained from P using the usual binary tournament selection, function SelectParentsByRank(). We remark that this operation eliminates all the individuals P_S such that $C(P_S) > B$. Then, in lines 6-7, we apply the evolutionary operators, that will be explained in Section 3.5, on the selected population of parents *Mate*. In line 8 the set U is constructed by simply merging the two populations P and *Mate*. Line 9 generates the set $F = (F_1, F_2, \ldots)$ that consists of the non-dominated fronts of the set U; this is computed by the standard crowded-comparison operator, function FastNonDominatedSort(). In the **foreach** loop, at lines 12-18, elements of the sets F_j are sorted according the the standard function CrowdingDistanceAssignment() and then inserted, if possible, in the new population P. Finally, in lines 19-21, for the set F_L that could not be inserted in its entirety, only the best solutions are inserted into the new population P.

The above is the standard NSGA-II strategy. Next we describe the specific choices that we have made

Figure 2: Portfolio structure.

for EvoPortfolio: *(1)* chromosome and gene representation in Section 3.2, *(2)* multi-objective evaluation function in Sections 3.3-**??**, *(3)* initial population procedure in Section 3.4, and *(4)* evolutionary operators in Section 3.5.

3.2. Chromosome and gene representation

The population in our evolutionary algorithm is made up of individuals (chromosomes) that are S-portfolios for a given input stock configuration $S = \{A_1, \ldots, A_N\}$. As we can see in Figure 2, each S-portfolio P_S is a vector of integers P_i, where each P_i is the part of P_S invested in A_i (see Section 2.1). In the context of evolutionary process, we say that P_i is a *gene* of the *chromosome* P_S.

3.3. The multi-objective evaluation function

EvoPortfolio uses two objective functions: a *yield* objective function f_y and a *risk* objective function f_r. The yield function $f_y(P_S)$ is defined as the overall yield of P_s (see Section 2.1):

$$f_y(P_S) = Y(P_S) = \sum_{i=1}^{N} P_i \times y_i$$

while the risk function $f_r(P_S)$ is defined as the overall risk of P_s (see Section 2.1):

$$f_r(P_S) = R(P_S) = \sum_{i=1}^{N} P_i \times r_i$$

The objective is to maximize $f_y(P_S)$ and to minimize $f_c(P_S)$. Notice that, in order to make uniform the type of objective functions in terms of maximization, we consider $f'_c(P_S) = \frac{1}{f_c(P_S)}$, which is a function to maximize. Thus, we have to maximize $f_y(P_S)$ and $f'_c(P_S)$).

Let's consider the example presented in Figure 2... So $f_y(P_S) = xxx$, $f_c(P_S) = yyy$, and $f'_c(P_S) = zzz$.

3.4. Initial population

We start from an initial population P of p_{size} individuals. We have done experiments using two approaches for the initial population:

1. **random individuals:** we obtain p_{size} chromosomes by selecting a random P_i for each A_i in the input stock configuration.
2. **base individuals:** we start with a *base individual* P_S and create p_{size} individuals equals to P_S; in this approach the problem becomes: given an initial portfolio what is the best portfolio we need to reach? as we will see, in this case, we also define a mechanism to return the best actions (investments) to undertake to move from an initial portfolio to the best portfolio.

We have run tests using both choices. The tests with start individuals initial populations provided better solutions. The data we report has been obtained using this approach.

3.5. Evolutionary operators

We remark that the objective functions for our specific problem are defined on sequence of genes, i.e., are calculated by checking the yield and risk features of the genes of a chromosome. This property allows us to locate for each chromosome, the areas in which the characteristic of the genes are not good. We use this aspect to guide the evolutionary operators by fitness information: the operators, instead of operating in randomly selected points of the chromosomes, carefully choose such points to be the "worst areas" of the chromosome. This approach differentiates EvoPortfolio from many evolutionary algorithms, but reflects a natural way of investing in the stock market. In fact, when a user invests money, usually intervenes in the worst points of his portfolio, buying or selling specific stocks, and evaluating the quantities of appropriate stocks to invest in.

It's common to fix the number of crossover points at a very low constant value of 1 or 2. We use a double-point crossover during the reproduction phase. In order to let the population evolve we apply the following evolutionary operators: yield crossover and mutation operators and risk crossover and mutation operators. Moreover, in order to implement the *elitism technique* we select at each generation those chromosomes that have at least one of the following characteristics: *(1)* there exists P_i such that $y_i \geq Y$, where Y is a parameter of the algorithm named *yield-threshold*, *(2)* there exists P_i such that $r_i \leq R$, where R is a parameter of the algorithm named *risk-threshold*.

Notice that since we use a multi-objective fitness function the best individuals are given by those in the Pareto front, that is the set of all non-dominated chromosomes. Thus, among all individuals in the Pareto front we choose the solution which has the best harmonic value. This is justified by the fact that, being forced to make a choice, the harmonic aspect can be considered more important.

In the following we describe the details of the crossover and mutation operators.

Yield and Risk Crossover. Let us consider the yield crossover. For each chromosome $P_S^1 = [P_1, \ldots, P_N] \in Mate$ selected for recombination, let (P_i, P_{i+1}) and (P_j, P_{j+1}), with $1 \leq i < j < N$, the two points of C_1 such that the values $f_y(P_i, P_{i+1})$ and $f_y(P_j, P_{j+1})$ are lowest among all pairs. These points are the *crossover points* considered for the yield recombination operator. Then, the algorithm looks at the chromosome $P_S^2 = [P'_1, \ldots, P'_N] \in Mate$ with the best value for $f_y(P'_i, P'_{i+1})$ and $f_y(P'_j, P'_{j+1})$ among all chromosomes in *Mate*. So we apply to P_S^1 and P_S^2 the yiels crossover operator to produce a new chromosome $P_S^3 = [P_1, \ldots, P_i, P'_{i+1}, \ldots, P'_j, P_{j+1}, \ldots, P_N]$.
The risk crossover works similarly (just substitute f_y with f_r).

Yield and Risk Mutation. Let us consider the yield mutation. For each chromosome $P_S \in Mate$ selected for mutation, we consider the pair (P_i, P_{i+1}) of genes with the worst value of $f_y(G_i, G_{i+1})$. Given a such pair (G_i, G_{i+1}), the yield mutation operator a new pair of genes (P'_i, P'_{i+1}) such that $f_y(P'_i, P'_{i+1}) > f_y(P_i, P_{i+1})$. Then a new chromosome P'_S is generated by replacing (P_i, P_{i+1}) with (P'_i, P'_{i+1}).
The risk mutation works similarly (just substitute f_y with f_r).

4. Related work

In this section,

5. Conclusion

Future works. We are currently planning

References

[1] C. M. Fonseca, P. J. Fleming, et al., Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization., in: Icga, Vol. 93, Citeseer, 1993, pp. 416–423.

[2] C. M. Fonseca, P. J. Fleming, An overview of evolutionary algorithms in multiobjective optimization, Evolutionary computation 3 (1) (1995) 1–16.

[3] J. Horn, N. Nafpliotis, D. E. Goldberg, A niched pareto genetic algorithm for multiobjective optimization, in: Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence, Ieee, 1994, pp. 82–87.

[4] N. Srinivas, K. Deb, Muiltiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary computation 2 (3) (1994) 221–248.

[5] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE transactions on evolutionary computation 6 (2) (2002) 182–197.

[6] R. Sarker, K.-H. Liang, C. Newton, A new multiobjective evolutionary algorithm, European Journal of Operational Research 140 (1) (2002) 12–23.