Cardinality Constrained Portfolio Optimization by Means of Genetic Algorithms

Ivica Martinjak
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
ivica.martinjak@fer.hr

***Abstract**. When applying the standard Markowitz mean-variance model on a real portfolio selection problem, we are faced with certain limitations like cardinality of chosen assets, discrete nature of trading variables etc. While the classical mean-variance model can be successfully solved by standard algorithms (quadratic programming), modelling an actual investment leads to NP-hard optimization problem. In such circumstances heuristic methods appear as the only way out.*

This paper aims at finding efficient evolutionary inspired algorithm for cardinality constrained portfolio optimization. Among developed algorithms which were able to solve problems with very many possible assets, the algorithm with hybrid crossover presents itself as the most effective. In order to make obtained results comparable, test sample was chosen from databases that serve as a benchmark for this problem class.

Keywords. portfolio optimization, rational investor, efficient frontier, heuristic algorithm, genetic algorithm

1 Introduction

On a market there is a variety of possible assets to invest in and an investor encounters decision problem what is the fraction of each asset that should be chosen to meet an investment goals. Although in reality there are many possible ways of investing, modern portfolio theory is based on a *rational investor*, who chooses these fractions in a manner to minimize risk and maximize an expected return on initial capital. The next question is how to measure risk and how to estimate expected return on a given asset – irrespective this being security or any another financial instrument. The most common approach of modelling risk and expected return was adopted by Markowitz [7] (*standard portfolio optimization problem*), and this model underlies all modern portfolio theory.

Even in case when the number of given assets is very small, the number of possible combinations of fractions $x_1, x_2, ..., x_N$ is huge, increasing extremely with number of assets *N*. Solving the classical portfolio optimization problem with hundreds and thousands of assets is very hard optimization problem, but still in a scope of many standard algorithms [10]. Nevertheless, when this model is used as a real-world tool, respecting constraints like cardinality makes this optimization problem severely difficult and classifies it into NP-hard problems. In such circumstances, deterministic optimization methods do not work efficiently and heuristic algorithms can be the only solution.

Genetic algorithms (GA) are search and optimization heuristic method, based on the natural evolution theory, which means that there are three basic rules underlying the algorithm: inheritance, variation and selection. The *solution space* of a treated problem is represented by the *population* – consisting of individuals (*chromosomes*). Each individual is an element of the solution space i.e. a candidate for the optimal solution (*global optimum*) of the problem. In each step of the algorithm chromosomes are evaluated by the *fitness function* and particular number of the best-ranked chromosomes (*parents*) is chosen to create the new generation of chromosomes (*children*) through *crossover* and *mutation*. The main role of the crossover is expected to be convergence towards optimum (since it is assumed that offspring resemble and even outperform, their parents), whereas mutation operator should provide an escape from the *local optimum*. This procedure can be repeated until the *evolution time* expired.

2 Standard mean–variance portfolio selection model

The classical mean-variance portfolio selection model determines the proportions of the initial capital x_i to be invested in particular asset *i*, so as to minimize the risk of the whole selected portfolio (represented by its variance) and maximize the expected return of the portfolio (estimated by history mean). More precisely, let a set of *N* assets is given, each having expected return r_i and standard deviation of history yield σ_i, and let μ_{ij} is a correlation coefficient between history yields of asset *i* and asset *j*. We denote a variance of the selected portfolio by *V*, whereas its expected return by *P*. Then, the classical mean-variance portfolio selection model is:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sigma_i\sigma_j\mu_{ij}x_ix_j = V \to \min \qquad (1)$$

subject to

$$\sum_{i=1}^{N} r_i x_i = P \qquad (2)$$

$$\sum_{i=1}^{N} x_i = 1 \qquad (3)$$

$$x_i \geq 0;\ \ i,j = 1,2,\ldots,N \qquad (4)$$

These equations tell us that for expected return *P* of a chosen portfolio, we are looking for such a combination of investment fractions $x_1, x_2, \ldots, x_N$ for which the risk (i.e. variance *V*) would be the least possible. On the other hand, the classical mean-variance portfolio selection model can be defined in a way that for a given variance *V*, expected return *P* should be maximized:

$$\sum_{i=1}^{N} r_i x_i = P \to \max, \qquad (5)$$

$i,j = 1,2,\ldots,N$.

The first definition determines an optimal portfolio for a given amount of risk, whereas the second definition gives an optimal portfolio for a particular amount of expected return. Since there are many possible risk figures, for a given set of *N* assets - as well as many possible expected returns, both of definitions lead us to the set of optimal portfolios. Furthermore, due to the fact that the definitions are equivalent, these two sets of optimal portfolios are the same; and can be graphically represented by a curve (*efficient frontier*) in the expected return–variance space.

Throughout this work, upper expressions shell be used in matrix form. Let $X = [x_1, x_2, \ldots, x_N]$ and $C = \begin{bmatrix} \sigma_1\sigma_1, & \sigma_1\sigma_2\mu_{12} \\ \sigma_2\sigma_1\mu_{12} & \sigma_2\sigma_2 \end{bmatrix}$, than eq. (1) can be written in a form

$$X \cdot C \cdot X^T = V \to \min \qquad (6)$$

In order to illustrate previously described ideas, let consider an example of two possible assets: assume we are forming portfolio of one equity with $r_1 = 9.5\%$ and $\sigma_1 = 0.018$, and another one having $r_2 = 12\%$ and $\sigma_2 = 0.026$. We shall suppose two different correlation coefficients between the assets, in order to get better impression of solution space of the problem (mean-variance coordinate system). Figure 1 shows sets of portfolios solving this two assets problem (less convex curve for $\mu_{12} = 0.1$, another one for $\mu_{12}' = -0.55$).

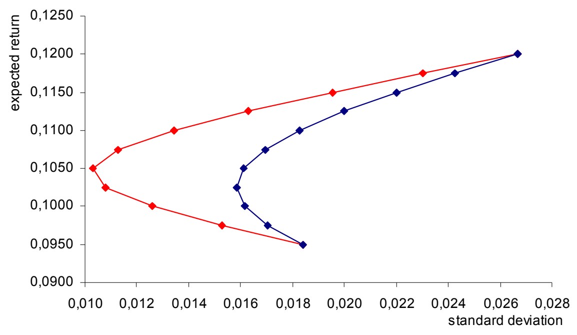

Figure 1. Efficient frontiers for the 2-assets portfolio

For example, an investor who is capable of taking a risk of 0.017 (in case $\mu_{12} = 0.1$), has two possible portfolios on disposal: one $[x_1, x_2] = [0.9, 0.1]$ with expected return 0.0975 and other $[x_1', x_2'] = [0.5, 0.5]$ with expected return 0.1075. Apparently, rational investor will choose combination $[0.5, 0.5]$, which has higher expected return.

3 Genetic algorithms for the standard portfolio optimization problem

In this paper, firstly we developed genetic algorithm with steady-state selection GAporto, for the standard portfolio optimization problem. The input parameters of the *N*-asset problem are expected return r_i, standard deviation σ_i and correlation μ_{ij} between every two assets *i*, *j*. Instead of seeking for a maximal possible expected return *P* for a given risk *V*, (or for a given P to minimize V) the algorithm will maximize the ratio $P/\sqrt{V}$; and a portfolio fulfilling this condition shall be called the optimal portfolio. This is a practice among researches in this field and we are going to keep it here making our results comparable with the others. Moreover, for the same purpose we use problem instances from known database "Operational Research Library" (OR-Library) [2] and from database which is prepared in work done by Cesarone, Scozzari, Tardella [3]. These databases are available on the internet (http://people.brunel.ac.uk/~mastjjb/jeb/info.html and http://w3.uniroma1.it/Tardella/homepage.html). From the first database we chose 5 problem instances which contain securities prices from capital markets Hang Seng, DAX 100, FTSE 100, S&P 100 and Nikkei; with number of possible assets in the portfolio from 31 to 225. The second database reveals securities prices from EuroStoxx50, FTSE 100, MIBTEL, S&P 500 and NASDAX; from 48 to 2196 prices in one portfolio. All these data were transformed in a way to serve only for scientific purpose. Table 1 shows a name of portfolio, the number of assets in portfolio N and the best known solution *best_known* (according to [2] and [3]) for the standard portfolio optimization model.

Table 1. Test sample

Portfolio	N	*best_known*
Hang Seng	31	0.210442
DAX 100	85	0.363785
FTSE 100	89	0.295636
S&P 100	98	0.319684
Nikkei	225	0.139380
EuroStoxx50	48	0.388722
FTSE 100	79	0.491485
MIBTEL	226	0.607284
S&P 500	476	0.513304
NASDAQ	2196	-

In the algorithm GAporto, every particular portfolio (candidate for the optimal one) is represented by chromosome consisting of N *genes*. Thus, genes represent the fractions $x_1, x_2, ..., x_N$ of the investment. Chromosome is implemented as a real N-length vector. We denote the number of chromosomes in the population by *POP*. The initial population is generated randomly, in a way that every chromosome picks up N values from the interval [0,1]. (One can notice that condition (3) is respected anyway. Namely, when dividing every element of the vector $[x_1, x_2, ..., x_N]$ by $\sum_{i=1}^{N} x_i$, the value of the ratio $P/\sqrt{V}$ will not be changed). In every iteration of the algorithm, *POP-M* chromosomes with the highest value of the fitness function are selected. These chromosomes serve as parents for the new generation – which is created by 2-point crossover, and which is written down on places of M worse candidates. After that, mutation operates so that every chromosome has probability p_m to be replaced by a random number from the interval [0,1]. The procedure will stop when given number of iteration *maxnumiter* is reached. The output of the algorithm is the optimal portfolio $[x_1, x_2, ..., x_N]$, its expected return P and variance V. Figure 2 gives the pseudocode of the described algorithm.

```
Algorithm GAporto
generate initial population P(0)
evaluate each individual in the population
t = 0
repeat
   select best-ranking individuals to reproduce
   create new generation through crossover
   and mutation (P(t) from P(t-1))
   evaluate each individual
   t = t +1
until (t<maxnumiter)
return best chromosome
```

Figure 2. Pseudocode of the algorithm GAporto

3.1 Input parameters optimization

When the algorithm was implemented, several series of experiments were done in a manner to optimize input parameters: mutation probability p_m, selection pressure M and the number of individuals in the population *POP*. First interesting observation, during this algorithm development phase, was stability of the algorithm. Namely, when series of several runs (let denote the number of runs in certain series by *numruns*) for a given problem instance were performed, differences among obtained results were very small (which is not always a case [9]). This is illustrated in Table 2, where the best, the worst and average solution of 10 algorithm runs for 85-assets portfolio can be seen.

Table 2. Results for 85-assets portfolio optimization (*POP*=100, *M*=90, p_m=0.015, *maxnumiter*=5000)

numruns	10
max	0.362789
min	0.362287
average	0.362521

Table 3 shows part of the results for p_m optimization; in most cases the best probability was between 0.015 and 0.020, so we chose figure 0.015 as a reference in our following experiments.

Except sensitivity to mutation probability, algorithm GAporto has shown very high sensitivity to the number of chromosomes M which will be replaced with the new generation (Table 4). Starting with M=50 and gradually increasing it, we realized that the best results can be reached around figure 90. To get better insight into GA behaviour, we were curious to see what is a characteristic selection pressure for some another problem class. As it was expected, obtained results indicate the rule that both mutation probability and selection pressure strongly depend on problem class whereas they slightly depend on problem instance.

Table 3. Results for p_m optimization (POP=100, M=90, *maxnumiter* =5000)

p_m	N=31	N=85
0.005	0.209995	0.362402
0.010	0.210181	0.362422
0.015	0.210147	**0.362473**
0.020	**0.210266**	0.362321
0.025	0.210259	0.361408

Table 4. Results for M optimization (POP=100, p_m=0.015, *maxnumiter*=5000)

M	N=31	N=85
70	0.210213	0.358837
80	**0.210252**	0.362243
86	0.210240	0.362233
90	0.210147	0.362473
92	0.210172	**0.362790**

3.2 Algorithm with hybrid crossover

Once algorithm GAporto was able to achieve results comparable with the best known results for specific portfolio instance, in rather small *maxnumiter*, we realized some ideas for additional enhancement of the algorithm (especially having in mind the fact that after a longer runtime there are many redundant chromosomes in the population).

Firstly, a comprehensive study of the influence of crossover type on the algorithm efficiency was done. Related tests included p-point crossover, $p \in (1,2,3,4,5)$ and uniform crossover. Through these tests, 2-point crossover presented itself as arguably the best performer among the crossover types – regardless of problem instance i.e. the number of assets in a portfolio. Furthermore, experiments with certain hybrid crossovers were done, eventually leading us to the improvement of the algorithm. In the case when parents are the same, one child is created based on its old genes - instead of regular 2-point crossover. More precisely, every gene of the child is replaced with new one – but with value from small range around value of old gene (Figure 3). It revealed that the range of 1% is the most appropriate for every analysed problem instance (i.e. constant a is equal to 0.01).

The algorithm with such a crossover performs better then GAporto for every problem instance from our sample and we shell call it GPporto_hc. Table 5 presents comparison between these two algorithms, showing average result from 10 algorithm runs for a given portfolio (*maxnumiter* was 5000, except in case of N=476 where it was 20000).

```
Procedure 2-pointCrHyb (int paA, paB, chA, chB)
break1, break2=Random[1,N]
if (break2<break1) repace its values
if ($paA \neq paB$)
  for j=0 to break1
    Chrom[chA][j]=Chrom[paA][j]
    Chrom[chB][j]=Chrom[paB][j]
  for j= break1+1 to break2
    Chrom[chA][j]=Chrom[paB][j]
    Chrom[chB][j]=Chrom[paA][j]
  for j=break2+1 to N
    Chrom[chA][j]=Chrom[paA][j]
    Chrom[chB][j]=Chrom[paB][j]
else
  for j=1 to N
     delta=Chrom[chA][j]·a·Random[0,1]·$(-1)^{\mathrm{Random}(1,2)}$
     Chrom[chA][j]= Chrom[chA][j]+delta
     if (Chrom[chA][j]<0) Chrom[chA][j]=0
end
```

Figure 3. Pseudocode of GAporto_hc crossover

Table 5. Comparison of GAporto and GAporto_hc (POP=100, M=90, p_m=0.015)

N	GAporto	GAporto_hc
31	0.210211	0.210396
85	0.362521	0.363112
89	0.294794	0.295161
98	0.319066	0.319386
225	0.070463	0.071304
48	0.388260	0.388601
79	0.490568	0.491112
226	0.545992	0.549582
476	0.363063	0.363574

Two examples of solution are as follows: obtained solutions for N=31 and N=48 in 5000 iteration (ratio $P/\sqrt{V}$ is equal to 0.210416 and 0.388660, respectively).

$[x_1, x_2, ..., x_{31}]$ = [0, 0.000065, 0, 0.000790, 0.572030, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.000133, 0, 0, 0.369456, 0, 0, 1.007923, 0, 0]

$[x_1, x_2, ..., x_{48}]$ = [0, 0.000155, 0, 0, 0.000090, 0, 0.000252, 0.122440, 0, 0.277020, 0, 0, 0, 0, 0, 0, 0, 0.353543, 0.376947, 0, 0, 0.000436, 0.190120, 0.000173, 0.000999, 0. 0.999054, 0, 0.483580, 0, 0, 0, 0, 0, 0, 0.000444, 0.000022, 0.000257, 0.193476, 0, 0, 0.023406, 0, 0, 0.000080, 0, 0.085790, 0.000067]

Input parameter optimization was done also for GAporto_hc and previously established conclusions on selection pressure and mutation probability were conformed. The only difference lies in optimal size of the population *POP*: in case of GAporto it holds the higher *POP* the better result, whereas in case of GAporto_hc this relation is more complex. Table 6 presents average results in 10 runs of GAporto_hc and shows that in some cases there are two maximums (*POP* of 150 and 300). Taking into account that bigger population size spend more time (and better measure of algorithm efficiency would be number of fitness function calculation) we stay at *POP* of 100 in our experiments. When optimizing the number of individuals in the population *POP* we also deal with several different combinations of *M* and p_m for certain figure of *POP* but without finding any generally better combination than *POP*=100, *M*=90, p_m=0.015.

Table 6. Results for *POP* optimization (*M*=90, p_m=0.015, *maxnumiter*=5000)

POP	*N*=31	*N*=79	*N*=225
50	0.210359	0.490872	0.066892
100	0.210396	0.491112	0.071304
150	0.210412	0.491151	0.071258
200	0.210335	0.490997	0.070582
300	0.210419	0.491328	0.069250

3.3 Analysis of algorithm efficiency

Neither with this optimization problem we could not avoid known general characteristic of the genetic algorithm: after quite fast founding the solution which is close to the optimal one, algorithm resumes its run with very slow increase of currently the best solution [8]. This fact is depicted on Figure 4: fitness function value of the best individuals in particular iteration can be seen for 31-assets portfolio. In the first run, GAporto_hc needs 483 iterations to reach solution better than 0.21 whereas in the second run 695 iterations have to be counted. Moreover, when we were looking for result better than 0.2104 the algorithm needed 3237 iterations to the first such a solution – which means that for result enhancement of 0.19%, 465% more time must be spent.

But this GA characteristic is not an obstacle in the cases of portfolios with small *N*, since only a moment is needed to solve them, i.e. to find solution equal to or better than the best known. (All experiments in this work were performed by PC Pentium Dual CORE CPU 2.50 GHz, 2GB RAM.) First more significant time the algorithm needed in case of *N*=225 and *N*=226 (to reach results better then 0.13 and 0.60, respectively). In the first case this time was 7955s (186642 iterations), whereas in the second case GAporto_hc needed 5740s (133454 iterations). Apart from standard portfolio optimization problem, all experiments with cardinality constrained portfolio optimization were done momentarily.

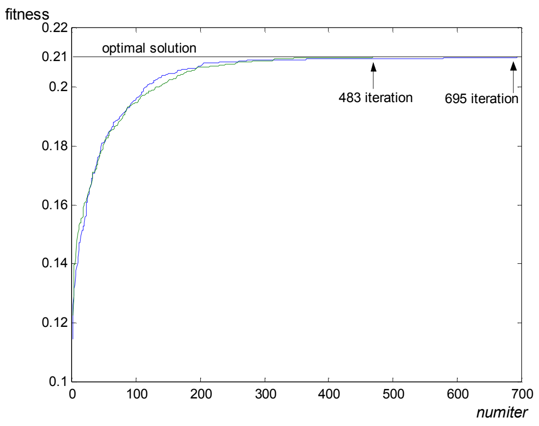

Figure 4. Progress of GAporto_hc towards the optimal solution for *N*=31 (*POP*=100, *M*=90, p_m=0.015)

4 Results for cardinality constrained portfolio

Applying portfolio selection model to a concrete investment, an essential limitation is the number of different assets *K*, which are selected into portfolio (cardinality constraint). Cardinality constrained portfolio optimization problem is defined by eq. (1) – (4) and eq. (7):

$$\sum_{i=1}^{N} sign(x_i) \leq K \qquad (7)$$

When developing the algorithm for this problem, we took GAporto_hc as a base. This time chromosome was designed as a complex structure, composed of two *K*-length parts. The first part contains chosen assets - which are represented by number $i \in (1,2,...,N)$, whereas the second part contains related investment fractions (Figure 5).

Parents

										fitness
13	24	28	8	18	0.74	0.06	0.45	0.86	0.73	0.13
11	25	18	10	9	0.82	0.65	0.60	0.50	0.07	0.12

Children

13	24	18	10	18	0.74	0.06	0.45	0.86	0.73	0.00
11	25	28	8	9	0.82	0.65	0.60	0.50	0.07	0.17

Figure 5. Two point crossover of GAccporto_hc, *N*=31

Two point crossover do not make any differences between these two components, but operates like there is 2*K*-length chromosome consisted of the same genes category.

After implementation of the algorithm, firstly we convinced ourselves that conclusions about input parameters optimization drawn in case of standard portfolio model were worth also in this issue (as Table

7, showing average results obtained in 10 algorithm runs, illustrates).

Table 7. Results for input parameters optimization (*maxnumiter*=5000, *K*=10)

M	p_m	*POP*	*N*=31	*N*=89
90	0.008	50	0.210441	0.294799
90	0.015	50	0.210442	0.294910
90	0.04	50	0.210442	0.294836
80	0.015	50	0.210442	0.294947
90	0.008	100	0.210442	0.294799
90	0.015	100	0.210442	0.294910
90	0.04	100	0.210442	0.294910
80	0.015	100	0.210432	0.294873

Several test series were performed, including cases for *K*=2,3,4,5,6,7,8,9,10,15,20. Extraction of obtained results one can see in Table 8 (*numruns*=1).

Table 8. Results for cardinality constrained portfolios (*POP*=100, *M*=90, p_m=0.015, *maxnumiter*=5000)

N	*K*=10	*K*=15	*K*=20
31	0.210442	0.210442	0.210442
85	0.363606	0.363795	0.363795
89	0.294947	0.295597	0.295597
98	0.314017	0.318654	0.319669
225	0.139373	0.139373	0.139370
48	0.388723	0.388723	0.388723
79	0.484447	0.491322	0.491490
226	0.565385	0.588873	0.600975
476	0.504032	0.511944	0.513214
2196	0.733023	0.865732	0.898707

In case of smaller problem instances, results for different *K* are often the same or very similar. For example, in cases *N*=31 and *N*=48, solutions which were obtained for *K*=10 are almost the same as those which can be achieved for *K*=15 or *K*=20. These two solutions are as follows.

$[x_4, x_5, x_8, x_9, x_{11}, x_{12}, x_{13}, x_{22}, x_{25}, x_{28}]$= [0.563552, 0, 0.316280, 0, 0, 0, 0, 0, 0.364388, 0.993132]

$[x_7, x_9, x_{17}, x_{18}, x_{22}, x_{26}, x_{28}, x_{38}, x_{41}, x_{46}]$= [0.129081, 0.288430, 0.367397, 0.392596, 0.198218, 1.045190, 0.501868, 0.202530, 0.024005, 0.090008]

When *K* is very small, like 2 or 3, obtained result (ratio between return and risk) is smaller than for higher amount of *K*.

4 Concluding remarks

This paper addresses the way in which genetic algorithms can be used to solve optimization problems related to the portfolio selection. Since the number of selected assets is an inevitable limitation when a real portfolio is formed, this work considers the cardinality constrained portfolio optimization problem. In addition to this NP-hard optimization problem, the developed algorithms were also solving the standard portfolio optimization problem.

Although solution space of both of problems increases enormously with the number of possible assets in a portfolio, developed genetic algorithms confirmed themselves as a very efficient heuristic method. More precisely, in case of the standard optimization problem, both of developed algorithms are very reliable for portfolios up to several hundreds assets. However, there is an impression that for the highest *N*'s – where the algorithms need more significant amount of runtime, additional researches are needed to clarify ability of GA. In case of the cardinality constrained portfolio optimization problem - which was our main objective because of its practical applications, our algorithm with hybrid crossover reached presented results in a few seconds.

According to our expectation, optimization of input parameters has shown that developed GA's are very sensitive to mutation probability and selection pressure. It is interesting that 2-point crossover present itself as the most effective among compared crossover operators, regardless on problem instance. Looking at demonstrated results from heuristic algorithm theory perspective, it can be said that achieved results contribute to the assumption that both mutation probability and selection pressure strongly depend on problem class whereas they slightly depend on problem instance.

References

[1] Bessis, Joel, **Risk Management in Banking**, Viley, Chichester, England, 1998.

[2] Beasley, J.E., **OR-Library: Distributing test problems by electronic mail**, Journal of the Operational Research Society, 1990, 41, pp. 1069-1072.

[3] Cesarone, F., Scozzari, A., Tardella, F., **Efficient Algorithms for Mean-Variance Portfolio Optimization with Hard Real-World Constraints**, 18th International AFIR Colloquium, Rome, 2008.

[4] Freedman, R., DiGiorgio R., **A Comparison of Stochastic Search Heuristics for Portfolio Optimization**, Proceedings of the Secong International Conference on Artificial Inteligence Application on Wall Street, 1993, pp. 149-151.

[5] HS Ryoo, **A compact mean-variance skewness model for large-scale portfolio optimization and its application to the NYSE market**,

Journal of the Operational Research Society, 2007, 58, pp. 505-515.

[6] Jorion, P., **Portfolio optimization in practice**, Financial Analysts Journal, January-February 1992, pp. 68-74.

[7] Markovitz, H. M., **Portfolio Selection**, Journal of Finance 7, 1952, 1, pp. 77-91.

[8] Martinjak, I., Golub, M., **Comparison of Heuristic Algorithms in Functions Optimization and Knapsack Problem**, IIS, Varaždin, 2006, pp. 413-418.

[9] Martinjak, I., Pavčević, M.O., **Modified Genetic Algorithm for BIBD Construction**, ITI, Cavtat, 2009, pp. 759-764.

[10] Streichert, F., Ulmer, H., Zell, A., **Evolutionary Algorithms and the Cardinality Constrained Portfolio Optimization Problem**, Proceedings of the International Conference on Operations Research, 2003.