Comparison of Multi-Agent Co-Operative Co-Evolutionary and Evolutionary Algorithms for Multi-Objective Portfolio Optimization

Rafał Dreżewski, Krystian Obrocki, Leszek Siwik

Department of Computer Science
AGH University of Science and Technology, Kraków, Poland
drezew@agh.edu.pl

Abstract. Co-evolutionary techniques makes it possible to apply evolutionary algorithms in the cases when it is not possible to formulate explicit fitness function. In the case of social and economic simulations such techniques provide us tools for modeling interactions between social or economic agents—especially when agent-based models of co-evolution are used. In this paper agent-based versions of multi-objective co-operative co-evolutionary algorithms are applied to portfolio optimization problem. The agent-based algorithms are compared with classical versions of SPEA2 and NSGA2 multi-objective evolutionary algorithms.

1 Introduction

Evolutionary algorithms are heuristic techniques which can be used for finding approximate solutions of global optimization problems. Evolutionary algorithms were also applied with great success to multi-modal and multi-objective problems (for example compare [2]), however in these cases some special mechanisms should be used in order to obtain good results. These are of course mechanisms specific for problems being solved but it seems that very important mechanisms in the case of multi-modal and multi-objective problems are the ones that maintain population diversity, because we are interested in finding not the single solution (as in the case of global optimization problems) but rather the whole sets of solutions.

Co-evolution is one of the mechanisms that can support maintaining of population diversity (see [10]). Another effect of applying co-evolutionary mechanisms is that we do not have to explicitly formulate the fitness function—we can just encode solutions in the genotypes and approximate fitness values for individuals on the basis of tournaments (competitive co-evolutionary algorithms) or co-operation (co-operative co-evolutionary algorithms).

Agent-based evolutionary algorithms are decentralized models of co-evolutionary computations. In fact two approaches are possible when we try to mix agent-based and evolutionary paradigms. In the first one agents are used to "manage" the evolutionary computations. In such approach each agent has the population of individuals inside of it, and this sub-population is evolved with the use of a standard evolutionary algorithm. Agents themselves can migrate within the computational environment, from one computational node to another, trying to utilize free computational resources.

The example of the second approach is *co-evolutionary multi-agent system (CoEMAS)* which results from the realization of (co-)evolutionary processes in multi-agent system (for example see [4]). In such systems agents "live" within the environment. All agents posses the ability to reproduce, they can compete for limited resources present within the environment, and die when they run out of resources. Multi-agent co-evolutionary algorithms based on CoEMAS model (utilizing different co-evolutionary mechanisms like predator-prey, host-parasite, co-operation, etc.) were already applied to multi-objective problems (for example see [7]) and to generating investment strategies ([6]).

Described above approaches can be mixed. For example, one can imagine the system in which agents serve as management layer, and individuals, which "live" within such agents are also agents. They can also migrate from one management agent to another and make independently all decisions.

Agent-based co-evolutionary systems have some distinguishing features, among which the most interesting seem to be: the possibility of constructing hybrid systems, in which many different computational intelligence techniques are used together within one coherent agent-based computational model, relaxation of computational constraints (because of decentralization of evolutionary computations), and the possibility of introducing new evolutionary operators and social relations, which were hard or impossible to introduce in the case of "classical" evolutionary computations.

The system presented in this paper uses agents for managing evolutionary computations. Additionally, agent-based co-operative co-evolutionary approach is adapted for solving the multi-objective problem of portfolio optimization. The results of experiments are used to compare proposed agent-based co-operative co-evolutionary algorithm, agent-based co-operative versions of well known SPEA2 and NSGA2 algorithms, and original versions of SPEA2 and NSGA2.

2 Agent-Based Co-Operative Co-Evolutionary System

The general architecture of the system presented in this paper was described with details in [5]. Here we rather concentrate on the details of algorithms used in the system. The system was implemented with the use of Java based framework jAgE ([1]). This framework is particularly suitable for implementing agent-based evolutionary algorithms because it provides all necessary elements like environment composed of computational nodes, agents, basic mechanisms for agent-agent and agent-environment interactions.

The system which we describe here has five implemented algorithms. Agent-based algorithms utilize agent layer for managing evolutionary computations. Three versions of agent-based co-evolutionary algorithms were implemented: co-operative co-evolutionary multi-agent algorithm (*CCEA-jAgE*), agent-based co-operative co-evolutionary version of NSGA2 algorithm (*CCNSGA2-jAgE*), and agent-based co-operative co-evolutionary version of SPEA2 algorithm (*CCSPEA2-jAgE*). Also two classical multi-objective evolutionary algorithms were implemented: NSGA2 and SPEA2 (details of these algorithms may be found in [2]).

In the **co-operative co-evolutionary multi-agent algorithm (CCEA-jAgE)**, which is based on the co-operative algorithm proposed in [9], there are computational agents

Algorithm 1. The first step of the aggregate agent

```
1  for a ← a_1 to a_n do                    /*a_i is the i-th computational agent*/
2  |  receive the initial population P_a^0 from agent a ; /*P_a^0 is the sub-population of
   |  agent a in step 0*/
3  end
4  C = aggregation of the solutions from P^0 ;        /*C is the set of complete
   solutions (co-operations) consisted of the individuals coming from
   different sub-populations*/
5  calculate the contribution of each of the individuals in the co-operation;
6  for a ← a_1 to a_n do
7  |  send the sub-population P_a^0 to agent a;
8  end
9  A^0 = choose the non-dominated solutions from C ;          /*A is the set of
   non-dominated solutions found so far*/
```

Algorithm 2. Step of the computational agent

```
1  receive sub-population P^t from aggregate agent ;   /*P^t is the sub-population in
   time t*/
2  compute the fitness of individuals from P^t on the basis of their contribution to the whole
   solution quality;
3  P^{t+1} ← ∅;
4  while P^{t+1} is not full do
5  |  select parents from P^t;
6  |  generate offspring from parents and apply recombination;
7  |  P^{t+1} = P^{t+1} + offspring;
8  end
9  mutate individuals from P^{t+1};
10 send P^{t+1} to aggregate agent;
```

which have individuals inside of them. Computational agents are located within the computational nodes of the jAgE platform—these nodes can be located on the same machine or on different machines connected with network. Agent-aggregate (which is the kind of a central point of the system) is responsible for the creation of complete solutions and maintaining the set of non-dominated solutions found so far.

The first step of aggregate agent is shown in alg. 1. After each iteration of the algorithm computational agents send their sub-populations to the aggregate agent for evaluation (the single step of computational agent is shown in alg. 2). Aggregate agent then chooses individuals that would form the complete solutions (see alg. 3). Next, the aggregate agent calculates contributions of each individual from each sub-population to the whole solution of the problem (alg. 4). This is done by calculating the values of all criteria for all solutions. Then aggregate sends back the subpopulations to their respective computational agents in order to compute the fitness values of individuals. Such an approach results in minimizing the communication overhead generated by agents sending communicates via network. Also, in such a way the aggregate agent is able

Algorithm 3. Step of the aggregate agent managing the computations

```
1  while stop condition is not fulfilled do
2      for a ← a_1 to a_n do
3          receive sub-population P_a^t from agent a;
4      end
5      for a ← a_1 to a_n do
6          P_a^{t+1} = select individuals for new generation from P_a^{t-1} ⋃ P_a^t;
7      end
8      C^{t+1} ← complete solutions formed from P^{t+1};
9      calculate the contribution of individuals coming from different species to the whole
       solution quality;
10     for a ← a_1 to a_n do
11         send the sub-population P_a^{t+1} to the agent a;
12     end
13     update the set of non-dominated solutions A^{t+1} with the use of C^{t+1};
14 end
```

to update the set of non-dominated solutions found so far—it inserts into the set all non-dominated solutions from the current population and then removes from this set all solutions dominated by the newly inserted ones.

CCNSGA2-jAgE—agent-based co-operative co-evolutionary version of NSGA2 algorithm—is possible to obtain via the proper configuration of the previously described algorithm. The fitness computation in agent-based co-evolutionary NSGA2 is realized with the use of non-dominated sorting and crowding distance metric (see [2]). The aggregate agent joins the populations of parents and offspring, and chooses (on the basis of elitist selection and within each sub-population separately) individuals which will form the next generation sub-population. This sub-population will be then used for the creation of co-operations (complete solutions).

In the case of **agent-based co-operative co-evolutionary version of SPEA2 algorithm (CCSPEA2-jAgE)** some modifications of the algorithms presented previously had to be introduced. SPEA2 uses additional external set of solutions during the process of evaluating individuals ([11]), so in the agent-based co-evolutionary version of SPEA2 algorithm each computational agent has its own, local, external set of solutions (lA) used during the fitness estimation. This set is also sent to the aggregate agent. The step of selecting individuals to the next generation sub-population may be now omitted by aggregate agent, because lA^t is the set of parents—and it is obtained from computational agent. In order to create the set of complete solutions C^t and compute contributions of the individuals to the quality of the complete solutions, the aggregates are created from the individuals coming from populations P^t and lA^t.

Computational agent updates the archive lA^{t+1} with the use of mechanisms from SPEA2 ([11]). Parents are selected from lA^{t+1} and children generated with the use of recombination operator are inserted into P^{t+1} (offspring population). Then mutation is applied to the individuals from set P^{t+1} and it is sent to aggregate agent together with the individuals from lA^{t+1}.

Algorithm 4. Calculating the contribution of individuals coming from different species to the whole solution quality

```
for species P_s ← P_0 to P_n do
    choose representatives r_s from P_s;
end
C ← ∅;
for species P_s ← P_0 to P_n do
    for individual i_s ← i_0 to i_N do
        c_pool ← ∅;
        for j ← 1 to |r_s| do
            x ← aggregation of i_s with the representatives of the other species;
            compute F(x);
            c_pool ← c_pool + {x};
        end
        x ← solution chosen from c_pool;
        C ← C + {x};
        F(x) is set as the contribution of individual i_s to the whole solution quality;
    end
end
return C
```

3 The Experiments

The algorithms presented in the previous section were preliminary assessed with the use of commonly used multi-objective test DTLZ functions ([3]). The results of these experiments are presented in [5]. Generally speaking, the results obtained with the use of agent-based algorithms (especially CCEA-jAgE) were comparable, and in the case of some problems better, than those obtained with the use of SPEA2 and NSGA2. In this section we will present the results of experiments with the problem of multi-objective portfolio optimization.

In all compared algorithms (CCEA, CCNSGA2, CCSPEA2, NSGA2 and SPEA2) the binary representation was used. One point crossover and bit inversion was used as genetic operators. As the selection mechanism tournament selection with elitism was used. The size of the population was set to 50. In order to minimize the differences between algorithms the values of crucial (and specific to each algorithm) parameters were obtained during preliminary experiments.

The results presented in this section include Pareto frontiers generated by the algorithms. Also, in order to better compare the generated results, hypervolume metric was used. Hypervolume (HV) metric ([2]) allows to estimate both the convergence to the true Pareto frontier as well as distribution of solutions over the whole approximation of the Pareto frontier. Hypervolume describes the area covered by solutions of obtained approximation of the Pareto frontier result set. For each found non-dominated solution, hypercube is evaluated with respect to the fixed reference point.

In the case of optimizing investing portfolio complete solution is represented as a p-dimensional vector. Each decision variable represents the percentage participation of

Algorithm 5. The algorithm (based on the one-factor Sharpe model) of computing the expected risk level and income expectation

1 Compute the arithmetic means on the basis of rate of returns;
2 Compute the value of α coefficient $\alpha_i = \overline{R_i} - \beta_i \overline{R_m}$;
3 Compute the value of β coefficient $\beta_i = \frac{\sum_{t=1}^{n}(R_{it}-\overline{R_i})(R_{mt}-\overline{R_m})}{\sum_{t=1}^{n}(R_{mt}-\overline{R_m})^2}$;
4 Compute the expected rate of return of asset i $R_i = \alpha_i + \beta_i R_m + e_i$;
5 Compute the variance of random index ${s_{e_i}}^2 = \frac{\sum_{t=1}^{n}(R_{it}-\alpha_i-\beta_i R_m)^2}{n-1}$;
6 Compute the variance of market index ${s_m}^2 = \frac{\sum_{t=1}^{n}(R_{mt}-\overline{R_m})^2}{n-1}$;
7 Compute the risk level of the investing portfolio $\beta_p = \sum_{i=1}^{p}(\omega_i \beta_i)$;
8 ${s_{e_p}}^2 = \sum_{i=1}^{p}(\omega_i^2 {s_{e_i}}^2)$;
9 $risk = \beta_p^2 {s_m}^2 + {s_{e_p}}^2$;
10 Compute the portfolio rate of return $R_p = \sum_{i=1}^{p}(\omega_i R_i)$;

i-th ($i \in 1 \ldots p$) share in the whole portfolio. The problem is described with details in [8] (in this paper the agent-based predator-prey algorithm was used to solve this problem). Below we will present only the most important issues.

During presented experiments Warsaw Stock Exchange quotations from 2003-01-01 until 2005-12-31 were taken into consideration. Simultaneously, the portfolio consists of the three or seventeen stocks quoted on the Warsaw Stock Exchange. As the market index WIG20 has been taken into consideration.

During experiments one-factor Sharpe model was used. This model was also used in [8] (in this work also comparison to other models and explanation why this particular model was used during experiments may be found). The algorithm (based on the one-factor Sharpe model) of computing the expected risk level and income expectation related to the portfolio of p assets is presented in alg. 5. The meaning of symbols used in the alg. 5, are as follows: p is the number of assets in the portfolio, n is the number of periods taken into consideration (the number of rates of return taken to the model), α_i,β_i are coefficients of the equations, ω_i is the percentage participation of i-th asset in the portfolio, e_i is random component of the equation, R_{it} is the rate of return in the period t, R_{mt} is the rate of return of market index in period t, R_m is the rate of return of market index, R_i is the rate of return of the i-th asset, R_p is the rate of return of the portfolio, ${s_i}^2$ is the variance of the i-th asset, ${s_{e_i}}^2$ is the variance of random index of the i-th asset, ${s_{e_p}}^2$ is the variance of the portfolio, $\overline{R_i}$ is arithmetic mean of rate of return of the i-th asset, $\overline{R_m}$ is arithmetic mean of rate of return of market index.

The goal of the optimization is to maximize the portfolio rate of return and minimize the portfolio risk level. The task consists in determining values of decision variables $\omega_1 \ldots \omega_p$ forming the vector $\Omega = [\omega_1, \ldots, \omega_p]^T$, where $0\% \leq \omega_i \leq 100\%$ and $\sum_{i=1}^{p} \omega_i = 100\%$ and $i = 1 \ldots p$ and which is the subject of minimization with respect of two criteria $F = [R_p(\Omega) * (-1), risk(\Omega)]^T$.

The Pareto frontiers obtained for 3 stocks problem after 5000 fitness function evaluations in typical experiment are presented in figures 1, 2, and 3a. The figure 3b shows the average values of HV metric from 15 experiments for all compared algorithms. In

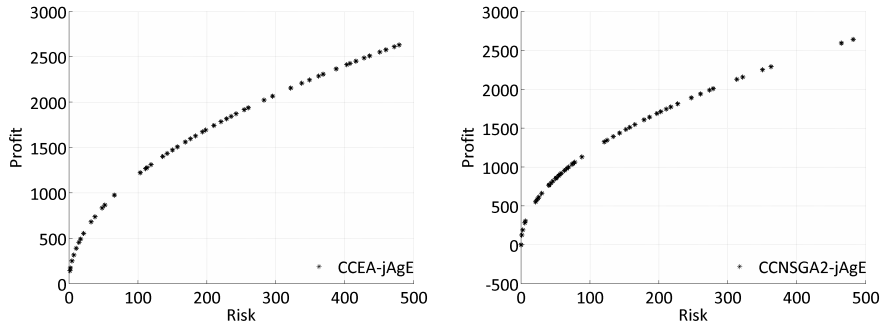


Fig. 1: Pareto frontiers obtained for 3 stocks problem after 5000 fitness function evaluations for CCEA (a) and CCNSGA2 (b)

this case (3 stocks) results are quite comparable for all implemented algorithms. Slightly worse results were obtained with the use of agent-based versions of SPEA2 and NSGA2 algorithms.

The Pareto frontiers obtained for 17 stocks problem after 25000 fitness function evaluations in typical experiment are presented in figures 4a–4e. In the figure 4f the average values of HV metric are presented (these are also average values from 15 experiments). In the case of the problem with 17 stocks the best results were obtained with the use of NSGA2 and SPEA2. When we look at the presented sample Pareto frontiers CCEA-jAgE algorithm formed quite comparable frontier—but the average value of HV metric was worser than in the case of NSGA2 and SPEA2. Agent-based versions of SPEA2 and NSGA2 decisively obtained worser results than other algorithms.

4 Summary and Conclusions

In this paper we have presented agent-based co-operative co-evolutionary algorithm for solving multi-objective problems. Also four other algorithms were implemented within the agent-based system: NSGA2, SPEA2 and agent-based co-operative co-evolutionary versions of these two state-of-the-art algorithms. In the paper [5] these algorithms were compared with the use of standard multi-objective test problems—DTLZ functions [3]. In this paper we have applied the system to the multi-objective problem of constructing optimal portfolio.

In the case of DTZL problems the winner was CCEA-jAgE algorithm—agent-based version of co-operative co-evolutionary algorithm (see [5]). In the case of optimal portfolio problem used in this paper the results are mixed. In the case of portfolio consisted of three stocks the results were rather comparable in the case of all algorithms—only agent-based versions of SPEA2 and NSGA2 algorithms obtained slightly worse results. In the case of seventeen stocks decisive winners are SPEA2 and NSGA2—especially

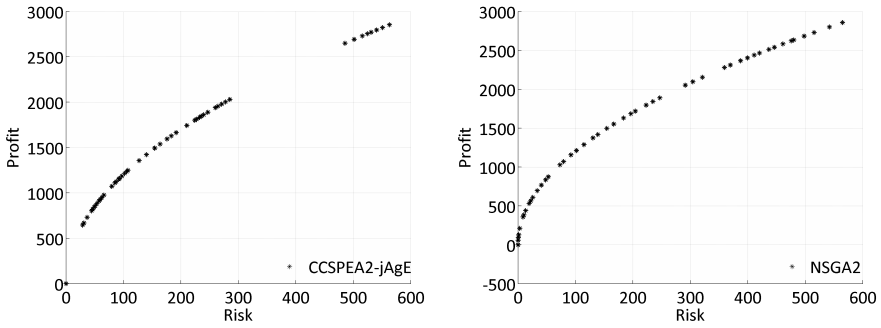


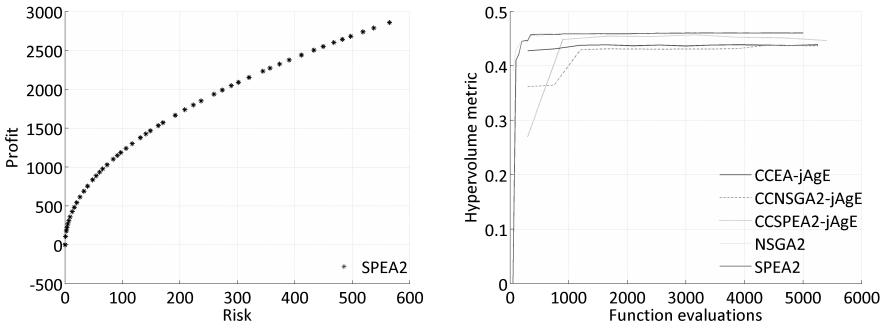
Fig. 2: Pareto frontiers obtained for 3 stocks problem after 5000 fitness function evaluations for CCSPEA2 (a) and NSGA2 (b)

when the values of HV metric are taken into consideration. Presented results lead to the conclusion that certainly more research is needed in the case of multi-objective agent-based techniques. But also it can be said that the results presented here (and in [5]) show that neither classical nor agent-based techniques can alone obtain good quality results for all kinds of multi-objective problems. We must carefully choose the right technique on the basis of the problem characteristics because there are no universal solutions. The algorithm that can obtain very good solutions for all types of multi-objective problems simply does not exist and we think that results presented here and in other our papers show this fact clearly.

When the future work is taken into consideration we can say that certainly presented agent-based algorithms will be further developed and tested on other multi-objective problems. Another direction of the research is (mentioned in section 1) the other way of merging multi-agent and evolutionary paradigms—the way in which agents are not used as the management layer but as the individuals that live, evolve and co-operate or compete with each other.

References

1. Agent-based evolution platform (jAgE). http://age.iisg.agh.edu.pl.
2. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.
3. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical report, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, 2001.
4. R. Dreżewski. Co-evolutionary multi-agent system with speciation and resource sharing mechanisms. *Computing and Informatics*, 25(4):305–331, 2006.
5. R. Dreżewski and K. Obrocki. Co-operative co-evolutionary approach to multi-objective optimization. Submitted to ICANNGA 2009.



(a) (b)

Fig. 3: Pareto frontier obtained for 3 stocks problem after 5000 fitness function evaluations for SPEA2 (a) and the values of HV metrics (b)

6. R. Dreżewski and J. Sepielak. Evolutionary system for generating investment strategies. In M. Giacobini, editor, *Applications of Evolutionary Computing*, volume 4974 of *LNCS*, pages 83–92, Berlin, Heidelberg, 2008. Springer-Verlag.
7. R. Dreżewski and L. Siwik. Agent-based co-operative co-evolutionary algorithm for multi-objective optimization. In L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, editors, *Artificial Intelligence and Soft Computing — ICAISC 2008*, volume 5097 of *LNCS*, pages 388–397, Berlin, Heidelberg, 2008. Springer-Verlag.
8. R. Dreżewski and L. Siwik. Co-evolutionary multi-agent system for portfolio optimization. In A. Brabazon and M. O'Neill, editors, *Natural Computation in Computational Finance*, pages 271–299. Springer-Verlag, Berlin, Heidelberg, 2008.
9. N. Keerativuttitumrong, N. Chaiyaratana, and V. Varavithya. Multi-objective co-operative co-evolutionary genetic algorithm. In J. J. Merelo, P. Adamidis, and H.-G. Beyer, editors, *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *LNCS*, pages 288–297. Springer-Verlag, 2002.
10. J. Paredis. Coevolutionary algorithms. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation, 1st supplement*. IOP Publishing and Oxford University Press, 1998.
11. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report TIK-Report 103, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, 2001.

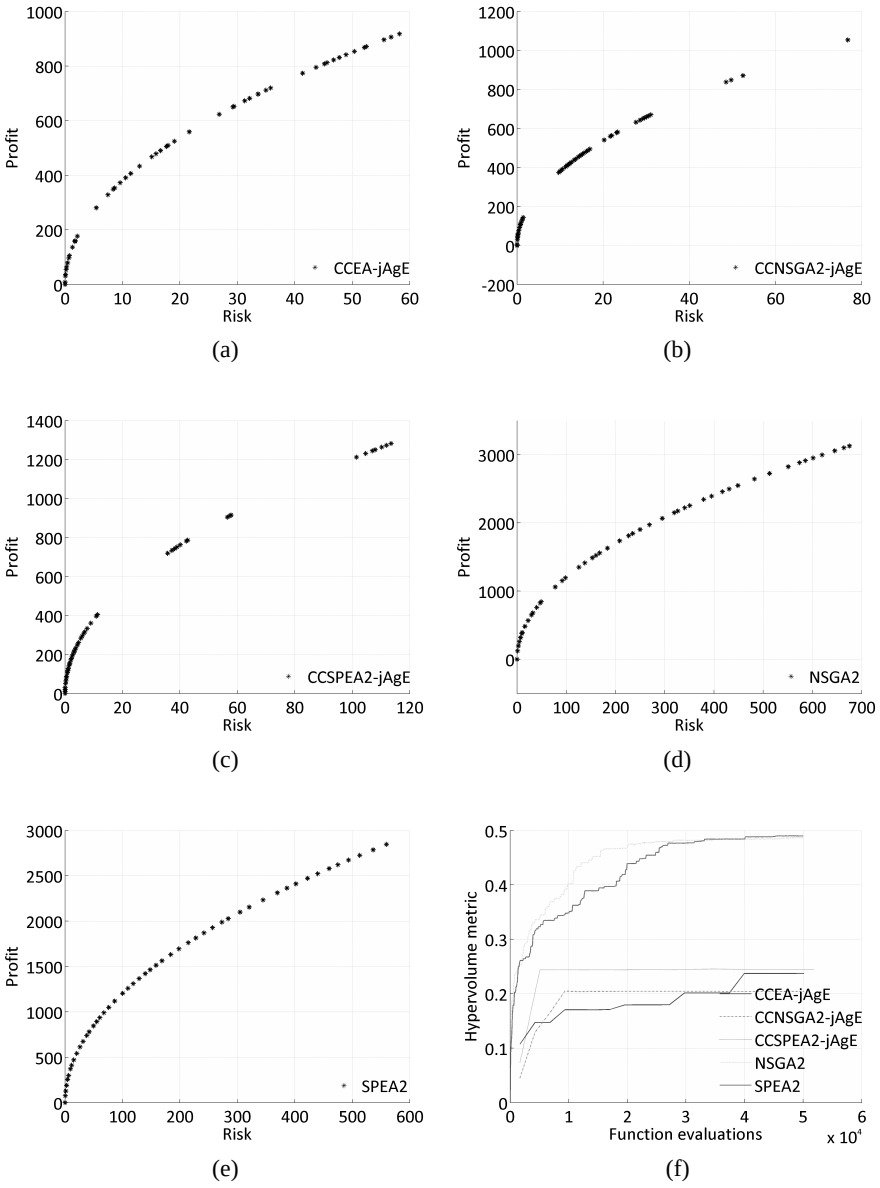


Fig. 4: Pareto frontiers obtained for 17 stocks problem after 25000 fitness function evaluations for CCEA (a), CCNSGA2 (b), CCSPEA2 (c), NSGA2 (d), SPEA2 (e) and the values of HV metrics (f)