Chapter 1
Agent-Based Co-Operative Co-Evolutionary Algorithms for Multi-Objective Portfolio Optimization

Rafał Dreżewski, Krystian Obrocki, Leszek Siwik

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 and economic agents—especially when agent-based models of co-evolution are used. In this chapter agent-based versions of multi-objective co-operative co-evolutionary algorithms are presented and applied to portfolio optimization problem. The agent-based algorithms are compared with classical versions of SPEA2 and NSGA2 multi-objective evolutionary algorithms with the use of multi-objective test problems and multi-objective portfolio optimization problem. Presented results show that agent-based algorithms obtain better results in the case of multi-objective test problems, while in the case of portfolio optimization problem results are mixed.

1.1 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 [1]), 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 a single solution (as in the case of global optimization problems) but rather the whole sets of solutions.

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

Co-evolution is one of the mechanisms that can support maintaining of population diversity (see [14]). 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 co-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 (see fig. 1.1). In such an 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 in a best way free computational resources.

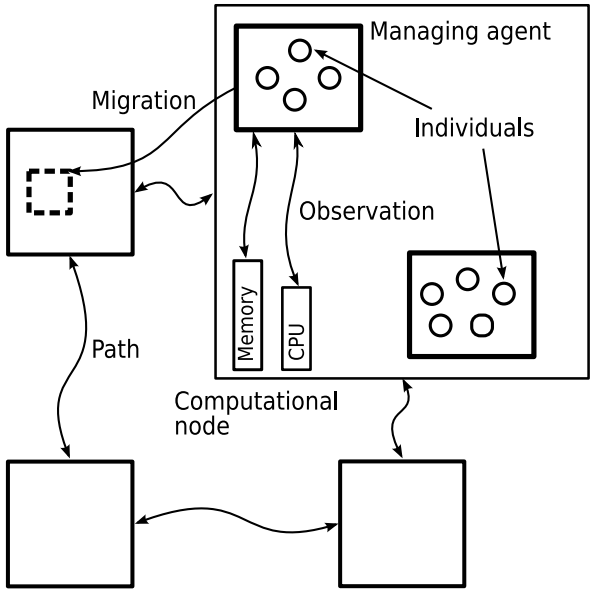


Fig. 1.1: Agent-based layer used for managing evolutionary computations

The example of the second approach is *co-evolutionary multi-agent system (Co-EMAS)* which results from the realization of co-evolutionary processes in multi-agent system (for example see [3, 4]). In such systems agents "live" within the environment (see fig. 1.2). 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.

In order to realize the selection process "better" (what means that they simply better solve the given problem) agents are given more resources from the environment (or from other agents) and "worse" agents are given less resources (or should give some of its resources to "better" agents). Such mechanisms result in decentralized evolutionary processes in which individuals (agents) make independently all their decisions concerning reproduction, migration, interactions with other agents, etc., taking into consideration conditions of the environment, other agents present within the neighborhood, and resources possessed.

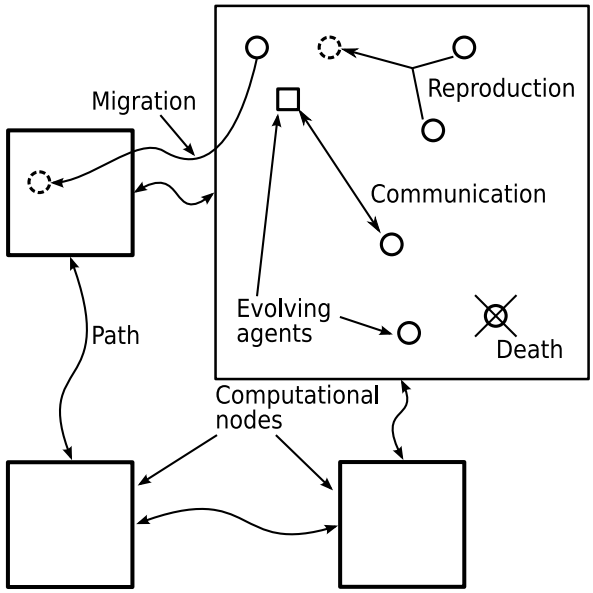


Fig. 1.2: Co-evolutionary multi-agent system—population of evolving agents

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 (see fig. 1.3). They can also migrate from one management agent to another and make independently all decisions (the system in which such approach was proposed is presented for example in [4]).

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 bio-inspired algorithms and techniques are used together within one coherent agent-based computational model,
- relaxation of computational constraints (because of the decentralization of evolutionary computations),

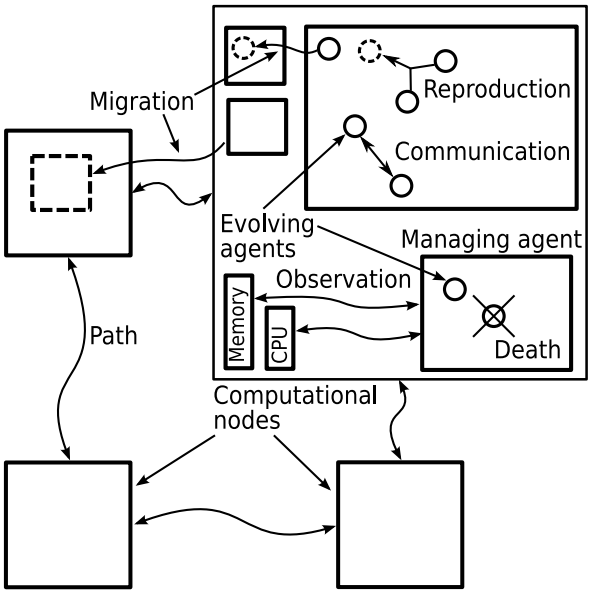


Fig. 1.3: Mixed approach—agent-based layer is used for managing computations and evolving individuals are agents

- the possibility of introducing new biologically and socially inspired operators or relations, which were hard or impossible to introduce in the case of "classical" evolutionary algorithms.

In the case of modeling and simulation of social and economic phenomena the model of co-evolutionary multi-agent system provides all necessary mechanisms like: agents, environment, agent-agent and agent-environment interactions needed for simulation of complex social systems. The basic model with biological (evolutionary) layer can be easily extended—social and economical layers can be added on the top of biological one. Thus we can construct artificial worlds and observe different emergent phenomena resulting from agents activities and interactions.

Multi-agent co-evolutionary algorithms based on CoEMAS model (utilizing different co-evolutionary interactions like: predator-prey, host-parasite, and sexual selection) were already applied to multi-objective problems (for example see [9], [7], [6]).

One of the first attempts of applying agent-based co-operative co-evolutionary approach to multi-objective optimization problems was presented in [8]. In the system presented in that paper the approach that uses agents as individuals living and evolving within the environment was used. There were several sub-populations (species) in the system. One criteria was assigned to each species. Agents competed for resources only within the species—there was no competition between agents that belonged to different species. Reproduction took place when the agent had enough

resources to perform it. The agent searched for a reproduction partner from one of the opposite species. As the multi-objective test problems ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 functions ([15]) were used. Co-operative co-evolutionary multi-agent system was compared to NSGA2 and SPEA2 algorithms. Obtained results showed that proposed algorithm initially allowed for obtaining better solutions, but with time classical algorithms—especially NSGA2—were the better alternatives. However, in the case of ZDT4 problem this characteristic was reversed—co-operative co-evolutionary multi-agent system finally obtained better results.

Agent-based co-evolutionary algorithms have also been applied to financial problems. Agent-based co-evolutionary algorithm with predator-prey interactions solving multi-objective portfolio optimization problem was presented in [9]. Co-operative co-evolutionary algorithm using genetic programming approach for generating investment strategies was described in [10]. These two systems were based on the first presented above approach to constructing agent-based co-evolutionary algorithms—individuals were agents, which competed for limited resources, could reproduce, migrate, and which could eventually die when they ran out of resources.

The system presented in this chapter uses agents for managing evolutionary computations (first of the presented above approaches of mixing agent-based systems and evolutionary algorithms). Additionally, agent-based co-operative co-evolutionary approach is adapted for solving the multi-objective problem of portfolio optimization. The results of experiments with multi-objective test problems and portfolio optimization problem 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.

The chapter is organized in the following way:

- In section 1.2 we will present the system and algorithms used in experiments: co-operative co-evolutionary multi-agent algorithm, agent-based co-operative co-evolutionary version of NSGA2 algorithm, and agent-based co-operative co-evolutionary version of SPEA2 algorithm.
- In section 1.3 there are results of experiments with the proposed algorithms presented. The problems used during experiments include commonly used multi-objective test functions: ZDT ([16]) and DTLZ ([2]), and multi-objective portfolio optimization problem.
- Conclusions and future work plans are presented in section 1.4.

1.2 Agent-Based Co-Operative Co-Evolutionary System

In the presented system co-operative co-evolutionary techniques were adapted to the demands of multi-objective problems and implemented with the use of mechanisms supported by the Java based framework jAgE ([12]). 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.

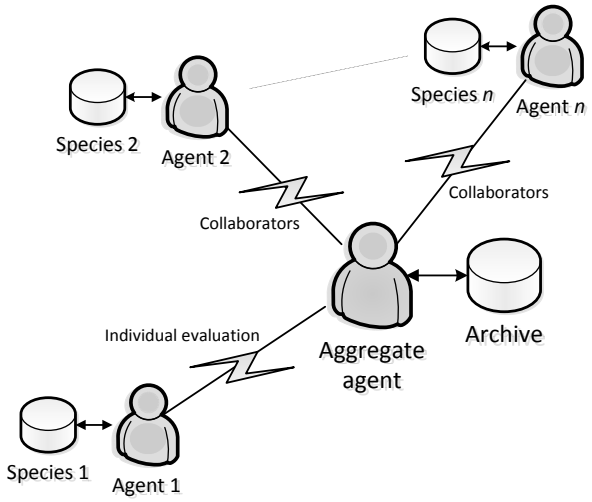


Fig. 1.4: The architecture of agent-based co-operative co-evolutionary system

Co-operative co-evolutionary approach can be easily parallelized because the interaction between individuals from different sub-populations takes place only during forming complete solutions and evaluating their fitness. In co-operative co-evolutionary algorithm computational nodes do not have to communicate very often—communication is needed only during evaluation of the solutions—thus the parallelization of computations can be realized effectively in the decentralized system (like jAgE), not only on parallel machines.

Because the representatives of each species (sub-populations) had to be aggregated (in order to form the complete solution) and also because of the necessity of storing the complete non-dominated solutions, the central computational node (agent-aggregate) was introduced (see fig. 1.4). Its tasks include forming complete solutions (composed of the representatives of each species) and evaluation of the solutions. It also maintains the set of non-dominated solutions found so far (the definition of domination relation and other issues connected with the Pareto approach to multi-objective optimization can be found for example in [1] or [9]). Each subpopulation is responsible only for the selected part of solution, and evolved by one computational agent.

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 [1]).

1.2.1 The Algorithms

1.2.1.1 Co-Operative Co-Evolutionary Multi-Agent Algorithm

In the **co-operative co-evolutionary multi-agent algorithm (CCEA-jAgE)**, which is based on the co-operative algorithm proposed in [13], there are computational agents 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 a kind of "central point" of the system) is responsible for the creation of complete solutions and maintaining the set of non-dominated solutions found so far.

Algorithm 1. The first step of the aggregate agent

1 **for** $a \leftarrow a_1$ **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** $a \leftarrow a_1$ **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 $P^{t+1} \leftarrow \emptyset$;
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;

In the first step of this algorithm each of the computational agents performs the initialization of its sub-population (which is associated with the selected part of the problem—in our case this is one decision variable). Aggregate agent waits for receiving all of the sub-populations. When it receives all sub-populations, it forms complete solutions and computes the contribution of individuals coming from each species (sub-populations) to the whole solution quality. Then the aggregate sends back all sub-populations and puts copies of all non-dominated solutions into the set of non-dominated solutions found so far (see alg. 1).

Each following step of computational agents (see alg. 2) begins with receiving of the sub-population from aggregate agent, then fitness of the individuals is computed. Next the selection of parents is performed, followed by the reproduction, recombination and mutation. At the end, the set of generated offspring is again sent to the aggregate agent.

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
```

Actions performed by the aggregate agent in the following steps start from checking whether the stop condition is fulfilled (see alg. 3). If yes, then the whole algorithm stops and the set of non-dominated solutions is the resulting Pareto frontier.

When the stop condition is not fulfilled then aggregate agent receives sub-populations from computational agents, and for each sub-population generates the set containing next generation of individuals (P^{t+1}) using individuals from previous generation of the given species and offspring sent by the given computational agent. Next the new set of complete solutions is generated on the basis of P^{t+1} and the contribution of individuals coming from different species to the whole solution quality is computed. Then the set of non-dominated solutions is updated (the new non-dominated solutions are inserted into the set and then all dominated solutions are removed from the set)—if the number of individuals in the set is greater than the maximal value then some individuals are removed on the basis of crowding al-

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
```

gorithm (individuals from the most "crowded" areas are removed in the first place). Next, sub-populations are sent back to computational agents.

The process of creating complete solutions (aggregating individuals) and computing the contribution of the given individual to the quality of the whole solution is made with the use standard co-operative co-evolutionary schema. Firstly representatives r_s of all species are chosen, and then for subsequent individuals i_s from subsequent species s the pool c_{pool} of complete solutions is created. For every solution from the pool (which is composed of the given individual i_s and representatives of all other species) the values of all criteria are computed. One solution is chosen from the pool and inserted into the set C of currently generated solutions. The vector of values $F(x)$ of the chosen solution is the measure of contribution of the given individual i_s to the quality of the solution (see alg. 4).

1.2.1.2 Agent-Based Co-Evolutionary Version of NSGA2 Algorithm with Co-Operative Mechanism (CCNSGA2-jAgE)

CCNSGA2-jAgE—agent-based co-operative co-evolutionary version of NSGA2 algorithm—is possible to obtain via proper configuration of the previously described algorithm (very similar solution was in fact applied in non-dominated sorting co-operative co-evolutionary genetic algorithm [11]).

As a result of integration of the previously described algorithm and NSGA2 ([1]) the agent-based co-operative version of NSGA2 was created. Thanks to the computed contribution of the given individual to the quality of the complete solution, the fitness computation in agent-based co-evolutionary NSGA2 is realized with the

use of non-dominated sorting and crowding distance metric (see [1]). Additionally, 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 used for the creation of complete solutions. The applied schema implies that N best (according to non-dominated sorting and crowding distance metric) individuals survive. Other parts of algorithm are realized in the same way as in the case of previously described agent-based co-operative algorithm.

1.2.1.3 Agent-Based Co-Evolutionary Version of SPEA2 Algorithm with Co-Operative Mechanism

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 done. It was caused mainly by the fact that SPEA2 uses additional external set of solutions during the process of evaluating individuals (compare [17]). In the described 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, along with the sub-population which is evolved by the given computational agent.

Algorithm 5. Step of the computational agent of CCSPEA2-jAgE algorithm

1 receive sub-population P^t and local external set of solutions lA^t from aggregate agent;
2 compute the fitness of individuals from P^t and lA^t on the basis of their contribution to the whole solution quality;
3 lA^{t+1} = environmental selection from $P^t \bigcup lA^t$;
4 $P^{t+1} \leftarrow \emptyset$;
5 **while** P^{t+1} *is not full* **do**
6 | select parents (using tournament selection) from lA^{t+1};
7 | generate offspring from parents and apply recombination;
8 | $P^{t+1} = P^{t+1} + \{offspring\}$;
9 **end**
10 mutate individuals from P^{t+1};
11 send P^{t+1} and lA^{t+1} to the aggregate agent;

First step of aggregate agent and computational agents is the same as in the case of CCEA-jAgE. Next steps of the algorithm of computational agents begin with receiving of the sub-population P^t and local external set of solutions lA^t from the aggregate agent (see alg. 5). On the basis of the contributions of the individuals to the quality of the complete solutions (computed by the aggregate agent), the fitness of individuals is computed. Next the archive lA^{t+1} is updated with the use of environmental selection mechanism adapted from SPEA2 algorithm ([17]). Parents are selected from lA^{t+1} and children generated with the use of recombination operator

Algorithm 6. Step of the aggregate agent managing the computations of CCSPEA2-jAgE algorithm

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

are inserted into P^{t+1} (offspring population). Then mutation is applied to the individuals from set P^{t+1} and this set is sent to the aggregate agent together with the individuals from lA^{t+1}.

In the case of aggregate agent, the changes include receiving and sending additional sets of individuals lA^t (see alg. 6). Due to the fact that lA^t is the set of parents, now the step of selecting individuals to the next generation sub-population may be omitted. 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. Finally all sub-populations are sent back to the proper computational agents and the set of non-dominated solutions is updated.

1.3 The Experiments

The algorithms presented in the previous section were preliminary assessed with the use of commonly used multi-objective ZDT ([16]) and DTLZ ([2]) test functions (detailed description of these test problems is presented in sections 1.3.1 and 1.3.2). Some of the results of these experiments were also 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 test problems better, than those obtained with the use of SPEA2 and NSGA2. In this section we will present results of selected experiments with test functions and the problem of multi-objective portfolio optimization.

1.3.1 ZDT test functions

Test problems designed by Zitzler, Deb and Thiele ([16]) represent typical difficulties faced while performing real life multi-objective optimization tasks. Each of the six problems consists in minimization of function:

$$\mathcal{T}(x) = \Big(f_1(x_1), f_2(x)\Big) \tag{1.1}$$

where $x = (x_1, ..., x_m)$, f_1 is a function of the first decision variable x_1 and f_2 is defined as:

$$f_2 = g(x_2, ..., x_m) \cdot h\Big(f_1(x_1), g(x_2, ..., x_m)\Big) \tag{1.2}$$

where g is a function of the remaining $m-1$ decision variables and the parameters of h are the function values of f_1 and g. Each of functions f_1, g and h is separately defined for every ZDT test problem. The number of decision variables m as well as their range of permissible values varies. ZDT problems are defined in the following way [16]:

- Test function $\mathcal{T}_1$ has a convex Pareto frontier formed with $g(x) = 1$ and is defined as follows:
$$\begin{cases} f_1(x_1) = x_1 \\ g(x_2, ..., x_m) = 1 + 9 \cdot \sum\limits_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \end{cases} \tag{1.3}$$
where $m = 30$ and $x_i \in \langle 0; 1 \rangle$.
- Test function $\mathcal{T}_2$ has a non-convex Pareto frontier formed with $g(x) = 1$:
$$\begin{cases} f_1(x_1) = x_1 \\ g(x_2, ..., x_m) = 1 + 9 \cdot \sum\limits_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2 \end{cases} \tag{1.4}$$
where $m = 30$ and $x_i \in \langle 0; 1 \rangle$.
- Test function $\mathcal{T}_3$ has a Pareto frontier composed of several non-continuous convex parts formed with $g(x) = 1$. The function is defined as follows:
$$\begin{cases} f_1(x_1) = x_1 \\ g(x_2, ..., x_m) = 1 + 9 \cdot \sum\limits_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right) \cdot \sin(10\pi f_1) \end{cases} \tag{1.5}$$
where $m = 30$ and $x_i \in \langle 0; 1 \rangle$.
- Test function $\mathcal{T}_4$ has 21^9 local Pareto frontiers. It is well suited for testing algorithm's capability of dealing with multi-modality. The true Pareto frontier is formed with $g(x) = 1$:

$$\begin{cases} f_1(x_1) = x_1 \\ g(x_2,...,x_m) = 1 + 10 \cdot (m-1) + \sum_{i=2}^{m} \left(x_i^2 - 10 \cdot \cos(4\pi x_i) \right) \\ h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \end{cases} \tag{1.6}$$

where $m = 10$ and $x_i \in \langle 0; 1 \rangle$.

- Test function $\mathcal{T}_5$ represents a problem with multiple deceptive local Pareto frontiers. The best of them is formed with $g(x) = 11$ while the global Pareto frontier is formed with $g(x) = 10$. The function requires binary representation of decision variables:

$$\begin{cases} f_1(x_1) = 1 + u(x_1) \\ g(x_2,...,x_m) = \sum_{i=2}^{m} v(u(x_i)) \\ h(f_1, g) = \frac{1}{f_1} \end{cases} \tag{1.7}$$

where $m = 11$, $x_1 \in \{0,1\}^{30}$ and $x_2,...,x_m \in \{0,1\}^5$. Function $u(x_i)$ gives the number of ones in the bit vector x_i and $v(u(x_i))$ is defined as follows:

$$v(u(x_i)) = \begin{cases} 2 + u(x_i) & , u(x_i) < 5 \\ 1 & , u(x_i) = 5 \end{cases} \tag{1.8}$$

- Test function $\mathcal{T}_6$ introduces difficulties based on non-uniformity of the search space. The density of solutions is decreasing while closing to the true Pareto frontier. The frontier is formed with $g(x) = 1$:

$$\begin{cases} f_1(x_1) = 1 - \exp(-4x_1) \cdot \sin^6(6\pi x_1) \\ g(x_2,...,x_m) = 1 + \left(9 \cdot \sum_{i=2}^{m} \frac{x_i}{m-1} \right)^{0.25} \\ h(f_1, g) = \frac{1}{f_1} \end{cases} \tag{1.9}$$

where $m = 10$ and $x_i \in \langle 0; 1 \rangle$.

1.3.2 DTLZ test functions

The main limitation of ZDT test functions is the use of two criteria only. Such simplification facilitates graphical illustration of solutions and their verification against true Pareto frontier localization. Scalable test problems proposed by Deb, Thiele, Laumanns and Zitzler ([2]) represent M-criteria optimization tasks. Each of the DTLZ problems consists in minimization of functions $f_1,...,f_m$. Due to space limitations, we define here only DTLZ1 problem, which will be used during presentation of the results of experiments [2]:

- DTLZ1 test problem has a linear Pareto frontier located on a hyperplane, which satisfies the condition $\sum_{m=1}^{M} f_m = 0.5$, and $11^k - 1$ local frontiers:

$$\begin{cases} f_1(\mathrm{x}) = \frac{1}{2} x_1 x_2 \cdots x_{M-1}(1 + g(\mathrm{x}_M)) \\ f_2(\mathrm{x}) = \frac{1}{2} x_1 x_2 \cdots (1 - x_{M-1})(1 + g(\mathrm{x}_M)) \\ \vdots \\ f_{M-1}(\mathrm{x}) = \frac{1}{2} x_1 (1 - x_2)(1 + g(\mathrm{x}_M)) \\ f_M(\mathrm{x}) = \frac{1}{2}(1 - x_1)(1 + g(\mathrm{x}_M)) \end{cases} \tag{1.10}$$

where $x_i \in \langle 0; 1 \rangle$ for $i = 1, 2, ..., n$ and $n = M + k - 1$ with suggested value of $k = |\mathrm{x}_M| = 5$.

1.3.3 Methodology of the Experiments

In all compared algorithms (CCEA, CCNSGA2, CCSPEA2, NSGA2 and SPEA2) the binary representation was used. One point crossover and bit inversion were 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.

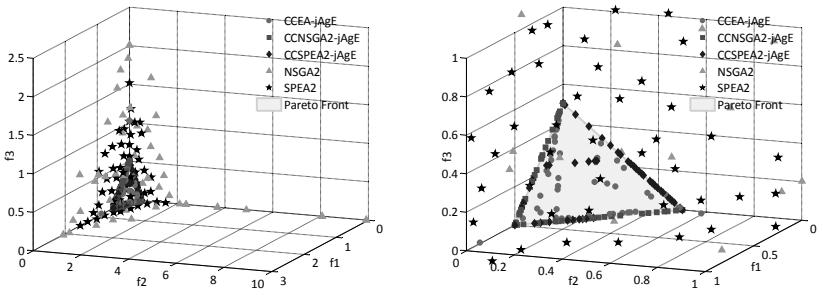
The results presented in this section include Pareto frontiers generated by the algorithms. Also, in order to better compare the generated results, hypervolume metric (HV) was used. Hypervolume metric ([1]) allows to estimate both the convergence to the true Pareto frontier as well as distribution of 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 order to evaluate hypervolume ratio, value of hypervolume for obtained set is normalized with hypervolume value computed for true Pareto frontier.

HV is defined as follows: $\mathrm{HV} = v\left(\bigcup_{i=1}^{N} v_i\right)$, where v_i is hypercube computed for $i - th$ found non-dominated solution, PF^* represents obtained approximation of the Pareto frontier and PF is the true Pareto frontier.

Values presented in the figures are averages from 15 runs of each algorithm against each test problem. Due to space limitations only selected Pareto frontiers and values of hypervolume metrics are presented.

1.3.4 Experiments with Multi-Objective Test Problems

Due to using three criteria in DTLZ problems it is possible to present the non-dominated solutions found by all compared algorithms. In the figure 1.5 results from runs of all algorithms against DTLZ1 function are presented. Its Pareto frontier was properly located by all agent-based algorithms. It is worth to mention though, that the solutions obtained by CCNSGA2-jAgE and CCSPEA2-jAgE algorithms were located on the edges of the frontier rather than on its surface. Solutions found by

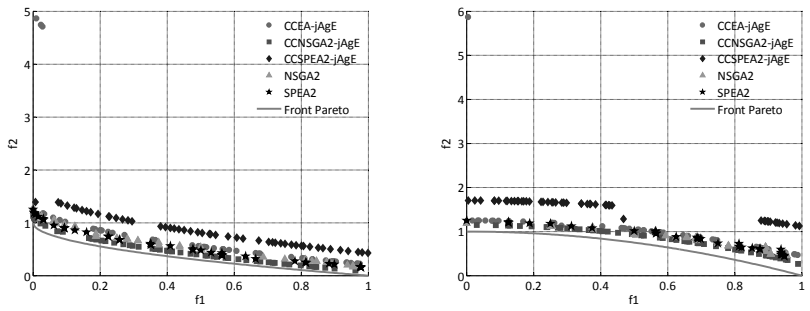


(a) The whole search domain (b) Magnification of the Pareto frontier

Fig. 1.5: Pareto frontiers obtained for DTLZ1 problem

NSGA2 and SPEA2 are all located at local Pareto frontiers (which are localized far away from the global frontier).

ZDT test problems are designed to use two criteria in order to facilitate presentation of the non-dominated solution sets in two dimensional space. In the figures 1.6-1.8 we present selected Pareto frontiers obtained during experiments.

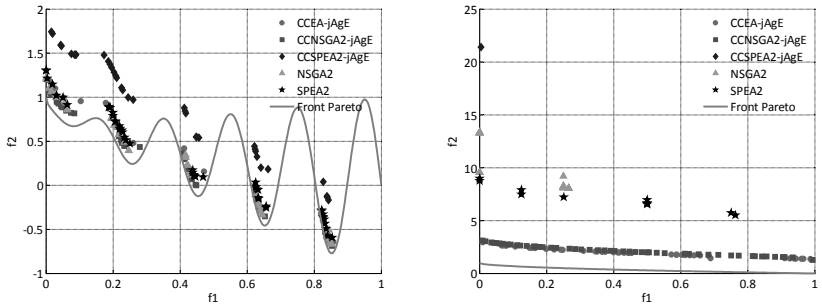


(a) (b)

Fig. 1.6: Pareto frontiers obtained for ZDT1 (a) and ZDT2 (b) problems after 5000 fitness function evaluations for all compared algorithms

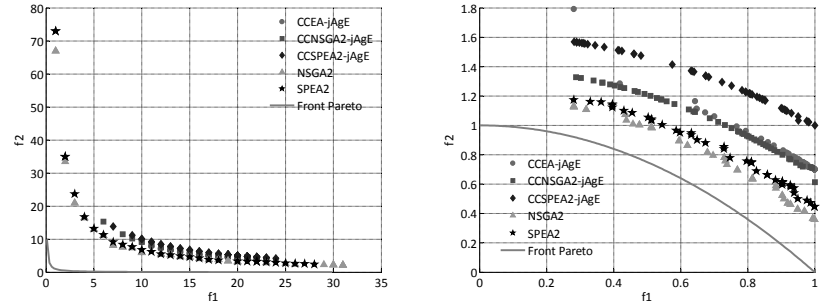
Multiple runs of the algorithms against test problems make it possible to present average values of hypervolume metric during experiments. In the figures 1.9-1.11 values of hypervolume metric are presented for all six ZDT test problems.

In the case of ZDT1, ZDT2 and ZDT3 test problems the quality of non-dominated sets obtained with the use of CCEA-jAgE, CCNSGA2-jAgE, NSGA2 and SPEA2



(a) (b)

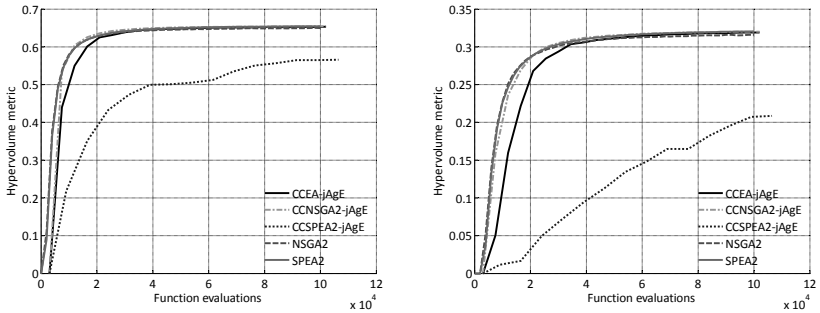
Fig. 1.7: Pareto frontiers obtained for ZDT3 (a) and ZDT4 (b) problems after 5000 fitness function evaluations for all compared algorithms



(a) (b)

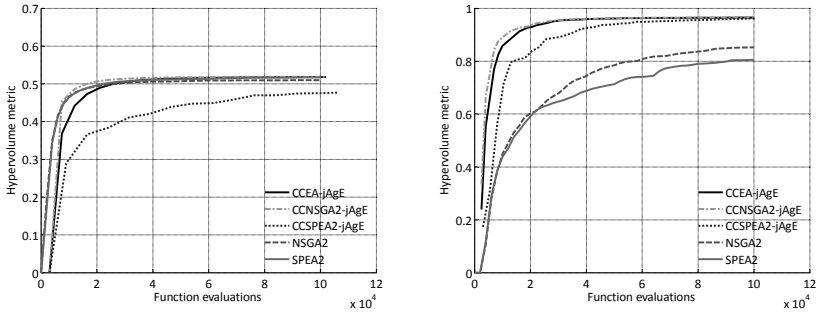
Fig. 1.8: Pareto frontiers obtained for ZDT5 (a) and ZDT6 (b) problems after 5000 fitness function evaluations for all compared algorithms

algorithms is comparable. CCSPEA2-jAgE performs noticeably worse in this case. As it can be seen in fig. 1.7b and fig. 1.10b agent-based co-evolutionary algorithms generate significantly better results for ZDT4 test problem than NSGA2 and SPEA2. On the contrary, in the case of ZDT5, the latter two produce solutions wider spread and located closer to the true Pareto frontier (see fig. 1.8a and fig. 1.11a). The quality of the solutions generated for ZDT6 problems is comparable in the case of all algorithms, though NSGA2 and SPEA2 show slightly faster convergence to the true Pareto frontier (see fig. 1.8b and fig. 1.11b).



(a) (b)

Fig. 1.9: Average values of hypervolume metric for ZDT1 (a) and ZDT2 (b) problems



(a) (b)

Fig. 1.10: Average values of hypervolume metric for ZDT3 (a) and ZDT4 (b) problems

1.3.5 Experiments with Multi-Objective Portfolio Optimization Problem

In experiments with multi-objective portfolio optimization problem complete solution is represented as a p-dimensional vector. Each decision variable represents the percentage participation of i-th ($i \in 1 \ldots p$) share in the whole portfolio. The problem is described with details in [9] (in that 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

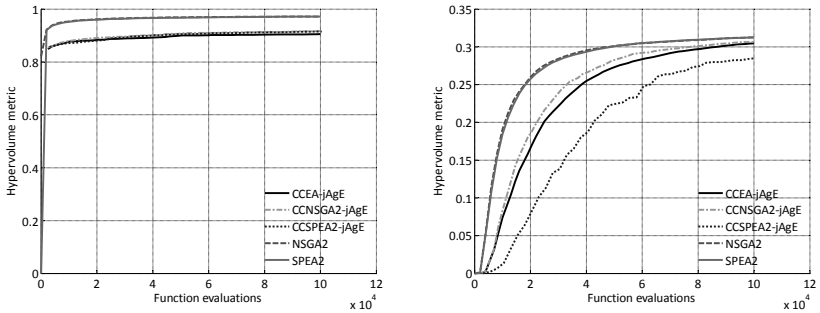


Fig. 1.11: Average values of hypervolume metric for ZDT5 (a) and ZDT6 (b) problems

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 [9] (in that 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. 7.

Algorithm 7. 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 $R_p = \sum_{i=1}^{p}(\omega_i R_i)$;

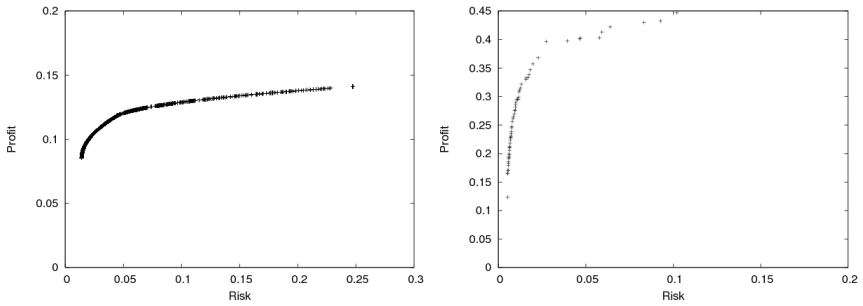
The meanings of the symbols used in alg. 7 are as follows:

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 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 the 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 $F = [R_p(\Omega) * (-1), risk(\Omega)]^T$.

Model Pareto frontiers for two cases (portfolios consisting of three and seventeen stocks set), which are the subject of analysis in the following section, are presented in fig. 1.12.

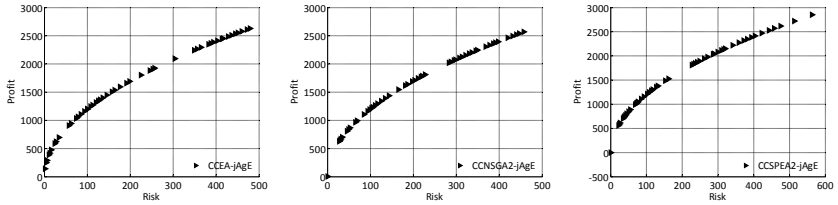


(a) (b)

Fig. 1.12: The model Pareto frontier obtained using utter review method for 3 (a) and 17 (b) stocks set

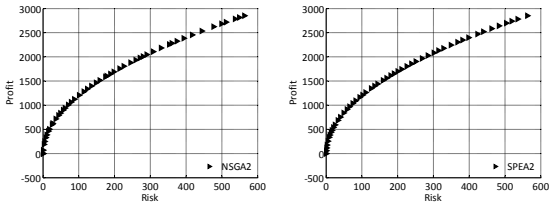
The Pareto frontiers obtained for 3 stocks problem after 2500 fitness function evaluations in typical experiment are presented in figures 1.13 and 1.14. Pareto frontiers obtained after 5000 fitness function evaluations are shown in figures 1.15 and 1.16.

The figure 1.21a shows the average values of HV metric from 15 experiments for all compared algorithms. In this case (3 stocks) results are quite comparable for



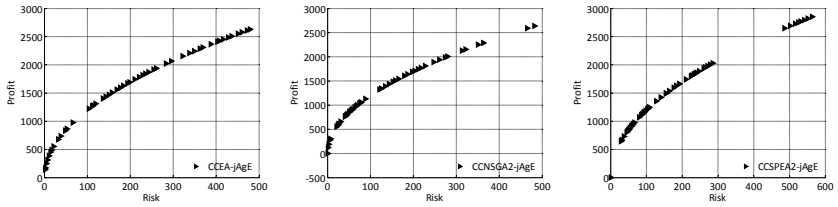
(a) (b) (c)

Fig. 1.13: Pareto frontiers obtained for 3 stocks problem after 2500 fitness function evaluations for CCEA (a), CCNSGA2 (b), and CCSPEA2 (c)



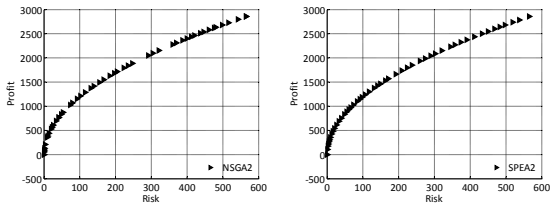
(a) (b)

Fig. 1.14: Pareto frontiers obtained for 3 stocks problem after 2500 fitness function evaluations for NSGA2 (a) and SPEA2 (b)



(a) (b) (c)

Fig. 1.15: Pareto frontiers obtained for 3 stocks problem after 5000 fitness function evaluations for CCEA (a), CCNSGA2 (b), and CCSPEA2 (c)

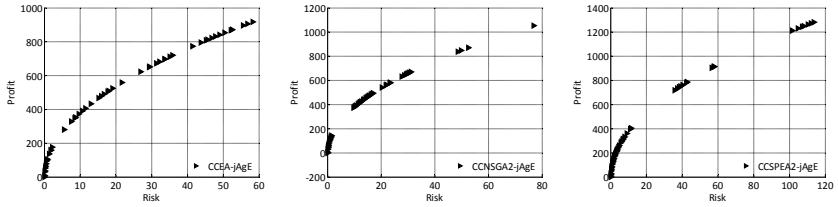


(a) (b)

Fig. 1.16: Pareto frontiers obtained for 3 stocks problem after 5000 fitness function evaluations for NSGA2 (a) and SPEA2 (b)

all implemented algorithms. Slightly worse results were obtained with the use of agent-based versions of SPEA2 and NSGA2 algorithms.

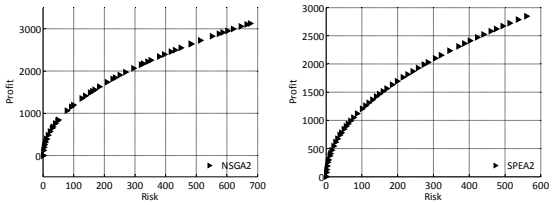
The Pareto frontiers obtained for 17 stocks problem after 25000 fitness function evaluations in typical experiment are presented in figures 1.17 and 1.18. Pareto frontiers obtained after 50000 fitness function evaluations are shown in figures 1.19 and 1.20.



(a) (b) (c)

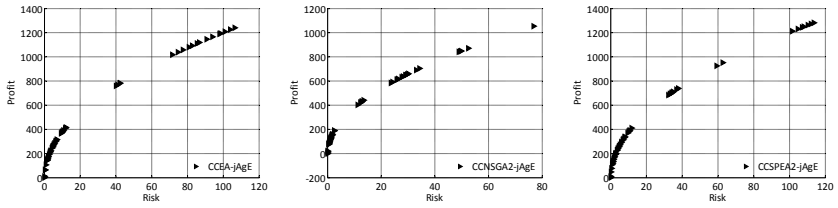
Fig. 1.17: Pareto frontiers obtained for 17 stocks problem after 25000 fitness function evaluations for CCEA (a), CCNSGA2 (b), and CCSPEA2 (c)

In the figure 1.21b 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 worse than in the case of NSGA2 and SPEA2. Agent-based versions of SPEA2 and NSGA2 decisively obtained worse results than other algorithms in this case.



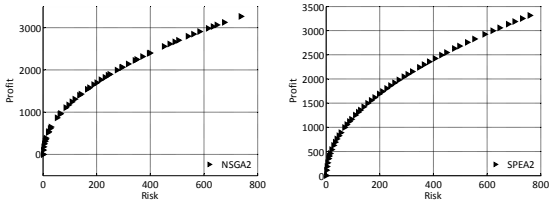
(a) (b)

Fig. 1.18: Pareto frontiers obtained for 17 stocks problem after 25000 fitness function evaluations for NSGA2 (a) and SPEA2 (b)



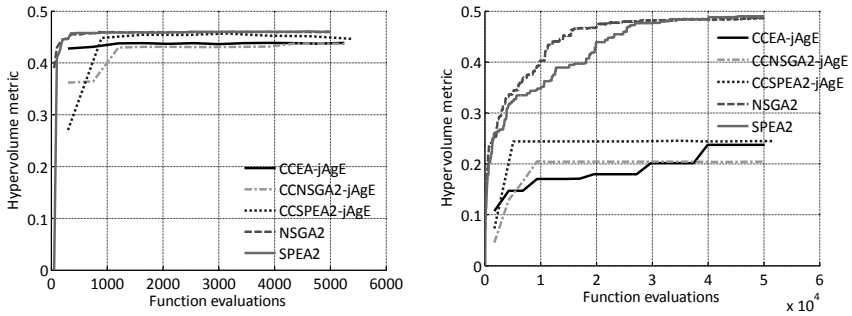
(a) (b) (c)

Fig. 1.19: Pareto frontiers obtained for 17 stocks problem after 50000 fitness function evaluations for CCEA (a), CCNSGA2 (b), and CCSPEA2 (c)



(a) (b)

Fig. 1.20: Pareto frontiers obtained for 17 stocks problem after 50000 fitness function evaluations for NSGA2 (a) and SPEA2 (b)



(a) (b)

Fig. 1.21: Values of HV metrics for 3 (a) and 17 (b) stocks problems

1.4 Summary and Conclusions

In this chapter 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. The algorithms were verified with the use of standard multi-objective test problems—ZDT ([16]) and DTLZ [2] functions, and the multi-objective problem of constructing optimal portfolio.

In the case of ZDT and DTZL problems the winner was CCEA-jAgE algorithm—agent-based version of co-operative co-evolutionary algorithm. In the case of optimal portfolio problem the results were mixed. In the case of portfolio consisted of three stocks the results of all algorithms were rather comparable—only agent-based versions of SPEA2 and NSGA2 algorithms obtained slightly worse results. In the case of seventeen stocks decisive winners were SPEA2 and NSGA2—especially when the values of HV metric were 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 1.1) 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. Beside the financial problems which we have already used in our research, like investment strategies generation or multi-objective portfolio optimization, we are also planning to use agent-based co-evolutionary approach in modeling and simulation of economical and social phenomena.

References

1. Deb K (2001) Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons
2. Deb K, Thiele L, Laumanns M, Zitzler E (2001) Scalable test problems for evolutionary multi-objective optimization. Tech. rep., Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology
3. Dreżewski R (2003) A model of co-evolution in multi-agent system. In: Mařík V, Müller J, Pěchouček M (eds) Multi-Agent Systems and Applications III, Springer-Verlag, Berlin, Heidelberg, LNCS, vol 2691, pp 314–323
4. Dreżewski R (2006) Co-evolutionary multi-agent system with speciation and resource sharing mechanisms. Computing and Informatics 25(4):305–331
5. Dreżewski R, Obrocki K (2009) Co-operative co-evolutionary approach to multi-objective optimization. In: Hybrid Artificial Intelligence Systems, Springer-Verlag, Berlin, Heidelberg, LNAI
6. Dreżewski R, Siwik L (2006) Co-evolutionary multi-agent system with sexual selection mechanism for multi-objective optimization. In: Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2006), IEEE
7. Dreżewski R, Siwik L (2006) Multi-objective optimization using co-evolutionary multi-agent system with host-parasite mechanism. In: Alexandrov VN, van Albada GD, Sloot PMA, Dongarra J (eds) Computational Science — ICCS 2006, Springer-Verlag, Berlin, Heidelberg, LNCS, vol 3993, pp 871–878
8. Dreżewski R, Siwik L (2008) Agent-based co-operative co-evolutionary algorithm for multi-objective optimization. In: Rutkowski L, Tadeusiewicz R, Zadeh LA, Zurada JM (eds) Artificial Intelligence and Soft Computing — ICAISC 2008, Springer-Verlag, Berlin, Heidelberg, LNCS, vol 5097, pp 388–397
9. Dreżewski R, Siwik L (2008) Co-evolutionary multi-agent system for portfolio optimization. In: Brabazon A, O'Neill M (eds) Natural Computation in Computational Finance, Springer-Verlag, Berlin, Heidelberg, pp 271–299
10. Dreżewski R, Sepielak J, Siwik L (2009) Classical and agent-based evolutionary algorithms for investment strategies generation. In: Brabazon A, O'Neill M (eds) Natural Computation in Computational Finance, vol 2, Springer-Verlag, Berlin, Heidelberg
11. Iorio A, Li X (2004) A cooperative coevolutionary multiobjective algorithm using non-dominated sorting. In: Deb K, Poli R, Banzhaf W, Beyer HG, Burke EK, Darwen PJ, Dasgupta D, Floreano D, Foster JA, Harman M, Holland O,

Lanzi PL, Spector L, Tettamanzi A, Thierens D, Tyrrell AM (eds) Genetic and Evolutionary Computation - GECCO 2004, Springer-Verlag, LNCS, vol 3102-3103, pp 537–548
12. jAgE—Agent-Based Evolution Platform (2009) http://age.iisg.agh.edu.pl
13. Keerativuttitumrong N, Chaiyaratana N, Varavithya V (2002) Multi-objective co-operative co-evolutionary genetic algorithm. In: Merelo JJ, Adamidis P, Beyer HG (eds) Parallel Problem Solving from Nature - PPSN VII, Springer-Verlag, LNCS, vol 2439, pp 288–297
14. Paredis J (1998) Coevolutionary algorithms. In: Bäck T, Fogel D, Michalewicz Z (eds) Handbook of Evolutionary Computation, 1st supplement, IOP Publishing and Oxford University Press
15. Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: methods and applications. PhD thesis, Swiss Federal Institute of Technology, Zurich
16. Zitzler E, Deb K, Thiele L (2000) Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2):173–195
17. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength pareto evolutionary algorithm. Tech. Rep. TIK-Report 103, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology