Co-operative Co-Evolutionary Approach to Multi-Objective Optimization

Rafał Dreżewski, Krystian Obrocki

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

Abstract. Co-evolutionary algorithms are evolutionary algorithms in which the given individual's fitness value estimation is made on the basis of interactions of this individual with other individuals present in the population. In this paper agent-based versions of co-operative co-evolutionary algorithms are presented and evaluated with the use of standard multi-objective test functions. The results of experiments are used to compare proposed agent-based co-evolutionary algorithms with state-of-the-art multi-objective evolutionary algorithms: SPEA2 and NSGA-II.

1 Introduction

Co-evolutionary algorithms [9] are particular branch of the evolutionary algorithms—robust and effective techniques for finding approximate solutions of global and multi-modal optimization problems. Co-evolutionary algorithms allow for solving problems for which it is impossible to formulate explicit fitness function because of their specific property—the fitness of the given individual is estimated on the basis of its interactions with other individuals existing in the population. The form of these interactions—co-operative or competitive—serves as the basic way of classifying co-evolutionary algorithms. Co-evolutionary interactions also promote the population diversity and introduce "arms races" among species (in the case of competitive interactions).

Many real-life decision making and optimization problems are multi-objective in nature. Usually we have to deal with many criteria, and making better the value of one of them usually means that other criteria values are worsening. There are quite many techniques of solving multi-objective problems. One of them is Pareto approach, in which we are interested in the whole set of so called "Pareto optimal" solutions (formal definition of multi-objective optimization problems, Pareto optimality, domination relation, and other basic notions, may be found for example in [2]). Evolutionary algorithms are techniques, which were recently applied with great success to solving multi-objective optimization problems—especially with the use of Pareto approach [2].

One of the problems which may occur during solving multi-objective problems with the use of evolutionary algorithms is the loss of population diversity. It is quite harmful in this case because the Pareto frontier would not be located properly—the algorithm (the population) would only locate selected parts of the frontier and in the case of multi-modal multi-objective problems (when many local Pareto frontiers exist [2]) there exists

the risk of locating local Pareto frontier instead of a global one. Co-evolution is one of the mechanisms that can be used in order to reduce the negative impact of the loss of population diversity.

The idea of integration of the multi-objective evolutionary algorithm and the co-operative co-evolutionary algorithm was proposed for the first time in [8]. The algorithm was verified with the use of standard multi-objective test problems. The experiments showed that the application of co-operative co-evolution leads to better results when compared to "classical" evolutionary approaches. Because of the principles of functioning of the co-operative co-evolutionary algorithm—multiple populations, which interact only during the fitness estimation—it is quite easy to implement its distributed version. First such attempt was made in distributed co-operative co-evolutionary algorithm (DCCEA) [10].

Agent-based evolutionary algorithms are a result of merging evolutionary computations and multi-agent systems paradigms. In fact two approaches to constructing agent-based evolutionary algorithms are possible. In the first one the multi-agent layer of the system serves as a "manager" for decentralized evolutionary computations. In the second approach individuals are agents, which "live" within the environment, evolve, compete for resources, and make independently all decisions (for example see [5]). Of course, all kinds of hybrid approaches are also possible.

In the case of first approach each agent holds inside its own sub-population of individuals and evolves them. Each agent also manages the computations, in such a way that it tries to minimize the communication delays, search for computational nodes which are not overloaded and migrates to them (with the whole sub-population of individuals), etc.

The paper starts with the presentation of agent-based co-operative co-evolutionary algorithms utilizing multi-agent layer as a "manager" for evolutionary computations. In the next section these algorithms are experimentally verified and compared to two state-of-the-art multi-objective evolutionary algorithms (SPEA2 and NSGA-II) with the use of commonly used multi-objective test problems.

2 Agent-Based Co-Operative Co-Evolutionary System for Multi-Objective Optimization

In this section the agent-based co-operative co-evolutionary system for multi-objective optimization is presented. In the described system agents are used rather as elements that manage the evolutionary computations, not as individuals that evolve themselves (see sec. 1 for the discussion of the possibilities of mixing agent-based systems and evolutionary computations). All versions of the algorithms were implemented with the use of agent-based evolutionary computations framework *jAgE* ([1])—this platform has all mechanisms and elements needed to implement agent-based evolutionary algorithms and it allows for the distributed computations. We will focus here on general system's architecture and implemented algorithms. Three versions of agent-based co-evolutionary algorithms are presented: co-operative co-evolutionary multi-agent algorithm (*CCEA-jAgE*), agent-based co-operative co-evolutionary version of NSGA-II

algorithm (*CCNSGA2-jAgE*), and agent-based co-operative co-evolutionary version of SPEA2 algorithm (*CCSPEA2-jAgE*).

In the presented system the co-operative co-evolutionary techniques were adapted to the demands of multi-objective problems and implemented with the use of mechanisms supported by the jAgE platform.

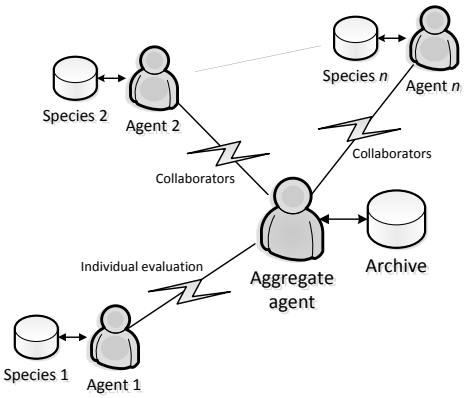

Fig. 1: The architecture of agent-based co-operative co-evolutionary algorithm

Because in the co-operative co-evolutionary approach the representatives of each species (sub-populations) have 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). 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. Each sub-population is responsible only for the selected part of the solution, and evolved by one computational agent.

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 the computations can be realized effectively in the decentralized system, not only on parallel machines.

Co-operative co-evolutionary multi-agent algorithm (CCEA-jAgE) is the agent-based and distributed version of multi-objective co-operative co-evolutionary algorithm based on algorithm proposed in [8].

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-

Algorithm 1. Step of the computational agent

receive P^t from aggregate agent ; `/*` P^t `is the sub-population in time` t`*/`
compute the fitness of individuals from P^t on the basis of their contribution to the solution quality;
$P^{t+1} \leftarrow \emptyset$;
while P^{t+1} *is not full* **do**
    select parents from P^t;
    generate offspring;
    apply recombination;
    $P^{t+1} = P^{t+1}$ + offspring;
end
mutate individuals from P^{t+1};
send P^{t+1} to aggregate agent;

Algorithm 2. Step of the aggregate agent

while *stopping condition is not fulfilled* **do**
    for $a \leftarrow a_1$ **to** a_n **do**
        receive P_a^t from agent a;
    end
    for $a \leftarrow a_1$ **to** a_n **do**
        P_a^{t+1} = select individuals from $P_a^{t-1} \bigcup P_a^t$;
    end
    $C^{t+1} \leftarrow$ complete solutions formed from P^{t+1};
    calculate the contribution of individuals coming from different species to the whole solution quality;
    for $a \leftarrow a_1$ **to** a_n **do**
        send P_a^{t+1} to the agent a;
    end
    update the set of non-dominated solutions A^{t+1} with the use of C^{t+1};
end

populations and puts copies of all non-dominated solutions to the set of non-dominated solutions found so far.

Following step of computational agents is presented in the alg. 1. Actions performed by the aggregate agent in the following steps are presented in alg. 2.

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 of 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. From the pool one solution is chosen 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.

As a result of integration of the previously described CCEA-jAgE algorithm and NSGA-II ([3]) the **agent-based co-operative version of NSGA-II (CCNSGA2-jAgE)** was created. CCNSGA2-jAgE is possible to obtain via the proper configuration of the CCEA-jAgE (very similar solution was in fact applied in non-dominated sorting co-operative co-evolutionary genetic algorithm [7]). Thanks to the computed contribution of the given individual to the quality of the complete solution, the fitness computation in agent-based co-evolutionary NSGA-II is realized with the use of non-dominated sorting and crowding distance metric (see [3]). 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.

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 [11]). 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.

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 the receiving of sub-population P^t and local external set of solutions lA^t from the aggregate agent. 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 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}.

In the case of aggregate agent, the changes include receiving and sending additional sets of individuals lA^t. 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.

3 The Experiments

The system presented in the previous section was experimentally verified with the use of commonly used test problems: DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, and DTLZ6 [4]. The system was also applied to multi-objective portfolio optimization problem—results can be found in [6]. The main goal of the experiments was to compare three agent-based co-operative co-evolutionary algorithms with two state-of-the-art multi-objective evolutionary algorithms: NSGA-II and SPEA2.

In all five compared algorithms (CCEA-jAgE, CCNSGA2-jAgE, CCSPEA2-jAgE, NSGA-II and SPEA2) the binary representation was used (32 bits per decision vari-

able). One point crossover and bit inversion was used as genetic operators. Probability of crossover was 0.9. The probability of mutation was $10/L$, where L is the length of the chromosome. Tournament selection with elitism was used in CCEA-jAgE, CCNSGA2-jAgE, NSGA-II algorithms and tournament selection without elitism in the case of CCSPEA2-jAgE and SPEA2. The size of the tournament was 3. The size of the population was set to 50. Maximal size of the set of non-dominated individuals was set to 50. Values presented in the figures are averages from 15 runs of each algorithm against each test problem. Due to space limitations only values of hypervolume metrics ([2]) are presented.

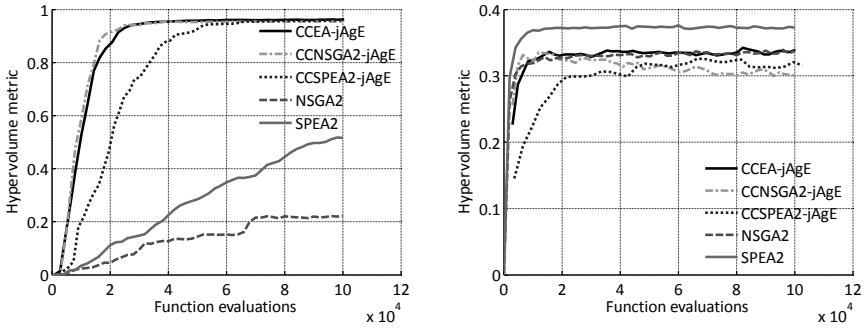

Fig. 2: Average values of hypervolume metric for DTLZ1 (a) and DTLZ2 (b) problems

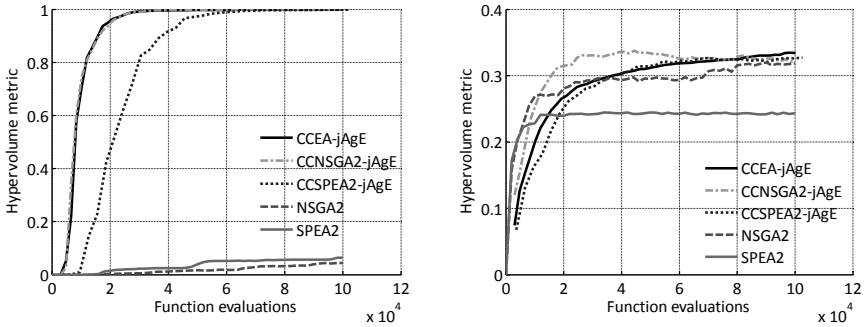

Fig. 3: Average values of hypervolume metric for DTLZ3 (a) and DTLZ4 (b) problems

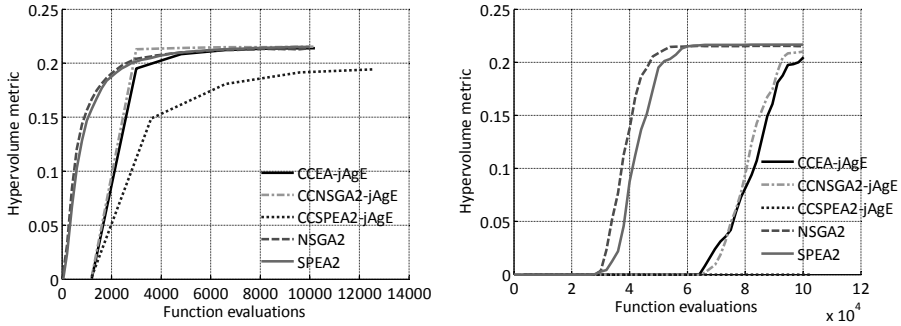

Fig. 4: Average values of hypervolume metric for DTLZ5 (a) and DTLZ6 (b) problems

In the figures 2-4 values of hypervolume metric are presented for all six test problems. In the case of DTLZ1 and DTLZ3 (fig. 2a and 3a) problems the best results were obtained with the use of proposed agent-based co-evolutionary algorithms. Also results of slightly slower in this case CCSPEA2-jAgE are better than those of SPEA2 and, the worst in this case, NSGA-II. In the case of DTLZ2 problem (fig. 2b) the best results were obtained by SPEA2 and slightly worse by CCEA-jAgE and NSGA-II. Results generated by CCNSGA2-jAgE and CCSPEA2-jAgE are less satisfying in this case. In the case of problem DTLZ4 (see fig. 3b) all algorithms generated comparable results, with the exception of SPEA2. The quality of the solutions generated for DTLZ5 problem (fig. 4a) is comparable in the case of all algorithms—only in the case of CCSPEA2-jAgE the average value of hypervolume metric is slightly lower than values for other algorithms. In the case of DTLZ6 function (fig. 4b) the Pareto frontier was not properly localized only by CCSPEA2-jAgE. The solutions obtained by other algorithms are of comparable quality, but NSGA-II and SPEA2 required about two times less fitness function evaluations to obtain such results.

4 Summary and Conclusions

In this paper agent-based co-operative co-evolutionary algorithm (CCSPEA2-jAgE) was proposed. In such system agents are used generally as the layer which manages the evolutionary computations. Thanks to the properties of co-operative co-evolutionary approach (interaction of individuals only at the stage of fitness evaluation of complete solutions) and properties of multi-agent approach, proposed algorithm was parallelized. The implementation was realized with the use of jAgE agent-based evolutionary framework, which allows for distributed computations. Also, within the same system, agent-based co-operative co-evolutionary versions of SPEA2 and NSGA-II algorithms were implemented.

Three proposed agent-based algorithms were experimentally verified with the use of DTLZ problems and compared to SPEA2 and NSGA-II algorithms. Presented results show that proposed CCSPEA2-jAgE obtained very satisfying results, comparable—and in the case of some problems even better—to those obtained by state-of-the-art SPEA2 and NSGA-II algorithms. Slightly less satisfying were the results obtained by proposed agent-based co-operative versions of SPEA2 and NSGA-II.

Future research will certainly include experiments with other multi-objective problems, not only with test functions but also with some real life problems. It will allow for additional verification of the proposed algorithms and will probably result in some improvements. On the other hand, different approach to agent-based realization of co-operative co-evolution will be further developed—the approach which utilizes agents as individuals living within the environment and independently forming co-operations (complete solutions).

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, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858. Springer, 2000.
4. 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.
5. R. Dreżewski. A model of co-evolution in multi-agent system. In V. Mařík, J. Müller, and M. Pěchouček, editors, *Multi-Agent Systems and Applications III*, volume 2691 of *LNCS*, pages 314–323, Berlin, Heidelberg, 2003. Springer-Verlag.
6. R. Dreżewski, K. Obrocki, and L. Siwik. Comparison of multi-agent co-operative co-evolutionary and evolutionary algorithms for multi-objective portfolio optimization. In *Applications of Evolutionary Computing*. Springer-Verlag, 2009.
7. A. Iorio and X. Li. A cooperative coevolutionary multiobjective algorithm using non-dominated sorting. In K. Deb and R. Poli, et al., editors, *Genetic and Evolutionary Computation - GECCO 2004*, volume 3102-3103 of *LNCS*, pages 537–548. Springer-Verlag, 2004.
8. 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.
9. 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.
10. K. C. Tan, Y. J. Yang, and T. H. Lee. A Distributed Cooperative Coevolutionary Algorithm for Multiobjective Optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, pages 2513–2520. IEEE Press, 2003.
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.