Function Optimization with Coevolutionary Algorithms

Franciszek Seredynski[1,2], Albert Y. Zomaya[3], and Pascal Bouvry[4]

[1] Polish -Japanese Institute of Information Technologies, Koszykowa 86, 02-008 Warsaw, Poland
[2] Institute of Computer Science of Polish Academy of Sciences, Ordona 21, 01-237 Warsaw, Poland
[3] School of Information Technologies, University of Sydney, Sydney, NSW 2006, Australia
[4] Institut Superieur de Technologie, Luxembourg University of Applied Science, 6, rue Coudenhove Kalergi, L-1359 Luxembourg-Kirchberg, Luxembourg

Abstract. The problem of parallel and distributed function optimization with coevolutionary algorithms is considered. Two coevolutionary algorithms are used for this purpose and compared with sequential genetic algorithm (GA). The first coevolutionary algorithm called a loosely coupled genetic algorithm (LCGA) represents a competitive coevolutionary approach to problem solving and is compared with another coevolutionary algoritm called cooperative coevolutionary genetic algorithm (CCGA). The algorithms are applied for parallel and distributed optimization of a number of test functions known in the area of evolutionary computation. We show that both coevolutionary algorithms outperform a sequential GA. While both LCGA and CCGA algorithms offer high quality solutions, they may compete to outperform each other in some specific test optimization problems.

1 Introduction

The use of evolutionary computation (EC) techniques to evolve solutions of both theoretical and real-life problems has seen a dramatic increase in popularity and success over last decade. The most popular and widely applied EC technique was a *sequential GA* ([5]) which computational scheme is based on a single population of individuals representing a single *species*. Development of parallel machines stimulated parallelization of a sequential GA and resulted in two parallel EC techniques known respectively as *island model* and *diffusion model* (see, e.g. [2]). These both models widely used today have been still exploring a notion of a single species, but individuals representing the species live in different subpopulations .

While these techniques are very effective in many applications, new more difficult problems were set. These problems (e.g. modeling economic phenomena such as a market) are in their nature distributed, i.e. can be seen as a number of independent interacting entities with own goals, where a global behavior can observed as the result of interactions. To meet these new

requirements, researches in the area of EC were looking for new more powerful paradigmes of natural processing. In the result *coevolutionary algorithms* based on modelling phenomena of coexistance of several species emerged [3] as a very promising area of EC.

In this paper we present a *competitive coevolutionary algoritm* called *LCGA*[8]), which is based on a game-theoretical model. The algorithm is parallel and distributed and can be interpreted as a multi-agent system with locally expressed (where it is possible) goals of agents and a global behavior of the system. We use our algorithm to solve the problem of *optimization of function* and compare it with another known from the literature *cooperative coevolutionary algorithm CCGA* ([7]) which is partially parallel and needs a global synchronization.

The paper is organized as follows. In the next section we shortly overview coevolutionary models existing in the area of EC. In Section 3 we describe test functions used in experiments. Sections 4 and 5 contain presentation of two coevolutionary algorithms *LCGA* and *CCGA* studied in the paper. Section 6 contains results of experiments conducted with use of coevolutionary algorithms. The last section contains conclusions of the paper.

2 Coevolutionary Genetic Algorithms

The idea of coevolutionary algorithms comes from the biological observations which shows that coevolving some number of *species* defined as collections of phenotypically similar individuals is more realistic than simply evolving a population containing representatives of one species. So, instead of evolving a population (global or spatially distributed) of similar individuals representing a global solution, it is more appropriate to coevolve subpopulations of individuals representing specific parts of the global solution.

A number of coevolutionary algorithms have been presented recently. The *coevolutionary GA* [6]) described in the context of the constraint satisfaction problem and the neural network optimization problem is a low level parallel EA based on a *predator-prey* paradigm. The algorithm operates on two subpopulations: the main subpopulation $P^1()$ containing individuals $\bar{x}$ representing some species, and an additional subpopulation $P^2()$ containing individuals $\bar{y}$ (another species) coding some constraints, conditions or test points concerning a solution $\bar{x}$. Both populations evolve *in parallel* to optimize a global function $f(\bar{x}, \bar{y})$.

The *cooperative coevolutionary GA* (CCGA) [7]) has been proposed in the context of a function optimization problem, and competetive coevolutionary algorithm called *loosely coupled GA* (LCGA) [8] has been described in the context of game-theoretic approach to optimization. These two coevolutionary algorithms are the subject of study presented in next sections. Another coevolutionary algorithm called *coevolutionary distributed GA* [4] was presented in the context of integrated manufacturing planning and scheduling

problem. It combines features of diffusion model with coevolutionary concepts.

3 Test Functions

In the evolutionary computation literature (see, e.g. [5]) there is a number of test functions which are used as benchmarks for contemporary optimization algorithms. In this study we use some number of such functions, which will be the subject of minimization. We use the following test functions:

- sphere model: a continuous, convex, unimodal function

$$f_1(x) = \sum_{i=1}^{n} x_i^2; \qquad x \in R^n, \tag{1}$$

  with $-100 \leq x_i \leq 100$, a minimum $x^* = (0, ..., 0)$ and $f_1(x^*) = 0$
- Rosenbrock's function: a continuous, unimodal function

$$f_2(\mathbf{x}) = \sum_{i=1}^{n} \left(100\left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2\right); x \in R^n, \tag{2}$$

  with $-2.12 \leq x_i \leq 2.12$, a global minimum
  $f_2(\mathbf{x}^*) = 0$ at $\mathbf{x}^* = (1, 1, \ldots, 1)$
- Rastrigin's function: a continuous, multimodal function

$$f_4(x) = \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i); \qquad x \in R^n, \tag{3}$$

  with $A = 10, \omega = 2 \cdot \pi - 5.12, \leq x_i \leq 5.12$,
  $f_4(x^*) = 0$ in $x^* = (0, 0, ..., 0)$
- Schwefel's function: a continuous, unimodal function

$$f_5(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2; \qquad x \in R^n, \tag{4}$$

  with $-100 \leq x_i \leq 100, f_4(x^*) = 0$ in $x^* = (0, 0, ..., 0)$
- Ackley's function: a continuous, multimodal function

$$f_6(x) = -20e^{(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})} - e^{(\frac{1}{n}\sum_{i=1}^{n} cos2\pi x_i)} + \tag{5}$$

  $+20 + e$
  with $-32 \leq x_i \leq 32, f_6(x^*) = 0$ in $x^* = (0, 0, ..., 0)$
- Griewank' function: a continuous, multimodal function

$$f_7(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1, \tag{6}$$

  with $-600 \leq x_i \leq 600, f_4(x^*) = 0$ in $x^* = (0, 0, ..., 0)$.

4 Competitive Coevolutionary Approach: Loosely Coupled Genetic Algorithms

Loosely coupled genetic algorithm (LCGA) [8,9] is a medium-level parallel and distributed coevolutionary algorithm exploring a paradigm of *competitive coevolution* motivated by noncooperative models of game theory. Local chromosome structures of LCGA are defined for each variable, and local subpopulations are created for them. Contrary to known sequential and parallel EAs, the LCGA is assumed to work in a distributed environment described by locally defined functions. A problem to be solved is first analyzed in terms of its possible decomposition and relations between subcomponents, expressed by a problem defined communication graph G_{com} called a graph of interaction.

In the case of functions like e.g. the Rosenbrock's function a decomposition of the problem, designing local functions $f_2^i(x_i, x_{i+1})$, and a graph of interaction (a local function assigned to an agent associated with the node i depends on a variable x_i associated with this node, and on the node $(i+1)$ with associated variable x_{i+1}) is straightforward (see, [1]).

Many real-life problems e.g. describing behavior of economic systems are naturally decentralized, or their models can be designed in such a way to decentralize their global criterion. When it is not possible, a communication graph G_{com} is a fully connected graph, and a global criterion becomes a local optimization criterion associated with each agent.

LCGA can be specified in the following way:

Step 1: for each agent-player create a subpopulation of his actions:
- create for each player an initial subpopulation of size *sub_pop_size* of player actions with values from the set S_k of his actions.

Step 2: play a single game:
- in a discrete moment of time each player randomly selects one action from the set of actions predefined in his subpopulation and presents it to his neighburs in the game.
- calculate the output of each game: each player evaluates his local payoff u_k in the game.

Step 3: repeat step 2 until *sub_pop_size* games are played.

Step 4: for each player create a new subpopulation of his actions:
- after playing *sub_pop_size* games each player knows the value of his payoff received for a given action from his subpopulation.
- the payoffs are considered as values of a local fitness function defined during a given generation of a GA; standard GA operators of selection, crossover and mutation are applied locally to the subpopulations of actions; these actions will be used by players in the games played in the next game horizon.

Step 5: return to step 2 until the termination condition is satisfied.

After initializing subpopulations, corresponding sequences of operations are performed *in parallel* for each subpopulation, and repeated in each generation. For each individual in a subpopulation a number of n_i (n_i-number of neighbors of subpopulation $P^i()$) of *random tags* is assigned, and copies of individuals corresponding to these tags are sent to neighbor subpopulations, according to the interaction graph. Individuals in a subpopulation are matched with ones that arrived uppon request from the neighbor subpopulations. Local fitness function of individuals from subpopulations is evaluated on the base of their values and values of arrived taggeted copies of individuals. Next, standard GA operators are applied locally in subpopulations. Coevolving this way subpopulations compete to maximize their local functions. The process of local maximization is constrained by neighbor subpopulations, sharing the same variables. As the result of this competitive coevolution one can expect the system to achieve some equilibrium, equivalent to a Nash point equilibrium in noncooperative models of game theory.

A final performance of the LCGA operated in a distributed environment is evaluated by some global criterion, usually as a sum of local function values in an equilibrium point. This global criterion is typically unknown for subpopulations (except the case when G_{com} is a fully connected graph), which evolve with their local criteria.

5 Cooperative Coevolutionary Approach: Cooperative Coevolutionary Genetic Algorithm

Cooperative coevolutionary genetic algorithm (CCGA) has been proposed [7] in the context of a function optimization problem. Each of N variables x_i of the optimization problem is considered as a species with its own chromosome structure, and subpopulations for each variable are created. A global function $f(\bar{x})$ is an optimization criterion. To evaluate the fitness of an individual from a given subpopulation, it is necessary to communicate with selected individuals from all subpopulations. Therefore, the communication graph G_{com} is fully connected.

In the initial generation of CCGA individuals from a given subpopulation are matched with randomly chosen individuals from all other subpopulations. A fitness of each individual is evaluated, and the best individual I^i_{best} in each subpopulation is found. The process of *cooperative coevolution* starts from the next generation. For this purpose, in each generation the following cycle consisting of two phases is repeated in a round-robin fashion. In the first phase only one current subpopulation is active in a cycle, while the other subpopulations are frozen. All individuals from an active subpopulation are matched with the best individuals of frozen subpopulations. A new better individual is found this way for each active subpopulation. In the second phase the best found individual from each subpopulation is matched with a single, randomly selected individual from other subpopulations. A winner individual

is a better individual from these two phases. When the evolutionary process is completed a composition of the best individuals from each subpopulation represents a solution of a problem.

6 Experimental Study

Both LCGA and CCGA algorithms were tested on the set of functions presented in Section 4. Results of these experiments were compared with the results of a sequential GA. In all experiments the accuracy of x_i was not worse than 10^{-6}. Experiments were conducted with number of variables $n = 5, 10, 20, 30$, but only results for $n = 30$ are reported in the paper. All algorithms run 200 generations.

The following parameters were set for LCGA: the size of subpopulation corresponding to given species was equal 100, p_m ranged for different functions from 0.001 (Rosenbrock's, Rastrigin's function) to 0.005, p_k ranged from 0.5 (Griewank's function), 0.6 (Rosenbrock's function) to 0.9. Ranking selection for Rastrigin's function and proportional selection for Ackley's function was used. For remaining functions a tournament selection with a size of tournament equal to 4 was used.

The following parameters were set for CCGA: the size of subpopulation corresponding to given species was equal 100, p_m ranged for different functions from 0.001 (Rosenbrock's, Rastrigin's, and Ackley's function) to 0.008, p_k ranged from 0.6 (Rosenbrock's function) to 1.0. Proportional selection for Rosenbrock's and Rastrigin function was used and ranking selection for Griewank's function was used. For remaining functions tournament selection with a size of tournament equal to 4 was used.

The following parameters were set for the sequential GA: the size of a global population corresponding to given species was equal 100, p_m ranged for different functions from 0.001 (Ackley's function) to 0.0095, p_k ranged from 0.5 (Griewank's function) to 0.95. Ranking selection for Rastrigin's and Griewank's function was used and tournament selection with a size of tournament equal to 3 was used for remaining functions.

Fig. 1 and 2 show results of experiments conducted with use of LCGA, CCGA and the sequential GA applied to minimize test functions presented in Section 4. Each experiment was repeated again 25 times. The best results in each generations of 20 the best experiments were averaged and accepted as results shown in following figures as a function of a number of generations.

One can easy notice that both coevolutionary algorithms LCGA and CCGA are better than the sequential GA in the problem minimization of function for all test functions used in the experiment. However, comparison of coevolutionary algorithms is not so straigthforward. For the test problem corresponding to the sphere model both coevolutionary algorithms present almost the same behavior (speed of convergence and the average of minimal values) for the number of variables from the range $n = 5$ to $n = 20$ (not

shown in the paper), with some better performance of LCGA. For $n = 30$ (see, Fig. 1a) the difference between both coevolutionary algorithms becomes more visible: LCGA maintenances its speed of convergence achieving the best (minimal) value after about 60 generations while CCGA is not able to find this value within considered number of 200 generations.

For the Rosenbrock's function CCGA shows slightly better performance that LCGA for all n, and this situation is shown in Fig. 1b for $n = 30$. One can see that this test function is really difficult for both algorithms. The opposite situation takes place for the Rastrigin's function. LCGA is distinctively better than CCGA for small values of n ($n = 5$) and slightly better for greater values of n. Fig. 1c illustrates this situation for $n = 30$.

For the Schwefel's function and $n = 5$ all three algorithms achieve the same minimal value. CCGA needs for this about 60 generations, LCGA needs about 110 generations and the sequential GA needs 200 generations. For $n = 10$ both coevolutionary algorithms are better than the sequential GA, but CCGA is slightly better than LCGA. For greater values of n CCGA outperforms LCGA (see, Fig. 2a). For the Ackley's function CCGA outperforms LCGA for all values of n (see, Fig. 2b) and the Griewank's function situation is similar Fig. 2c).

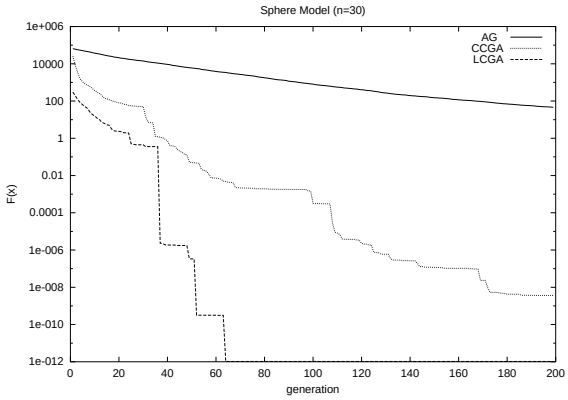
7 Conclusions

Results of ongoing research on the development of parallel and distributed evolutionary algorithms for function optimization have been presented in the paper. Coevolution - a new very promising paradigm in evolutionary computation has been chosen as an engine for effective parallel and distributed computation. Two coevolutionary algorithms based on different phenomena known as *competition* (LCGA) and *cooperation* (CCGA) were studied.

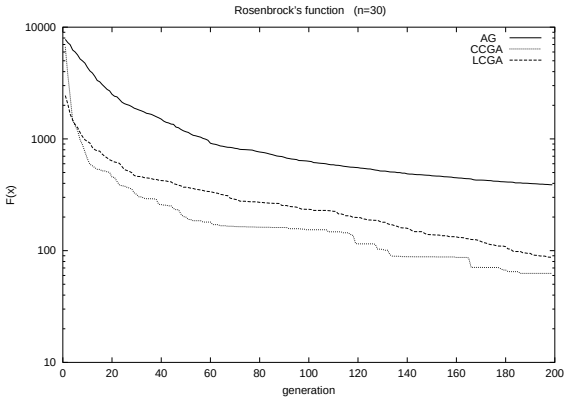
LCGA presents fully parallel and distributed coevolutionary algorithm in which subpopulations, using game-theoretic mechanism of competition, act to maximize their local goals described by some local functions. The competition between agents leads to establishing some equilibrium in which local goals cannot be more improved, and at the same time some global goal of the system is also achieved. The global state of the system (a value of a global goal) is not directly calculated, but is rather observed. To achieve this global goal no coordination of agents is required.

CCGA is partially parallel and centralized coevolutionary algorithm in which subpopulations cooperate to achieve a global goal. In a given moment of time only one subpopulation is active while the other subpopulations are frosen. The global goal of the system is at the same time a goal of each subpopulation. To evaluate a global goal a coordination center needs to communicate with each subpopulation to know a current local solution.

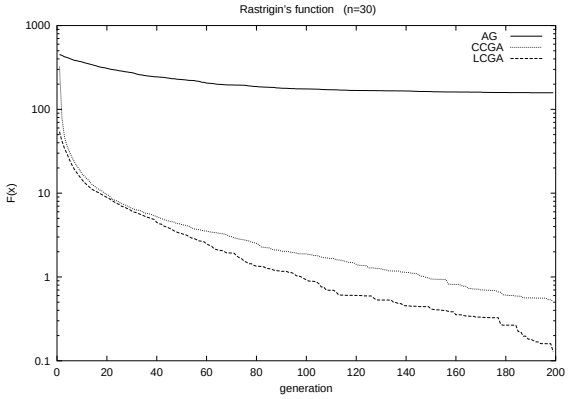
Results of experiments have shown that the LCGA is an effective optimization algorithm for problems where the global goal of the system is the



a)

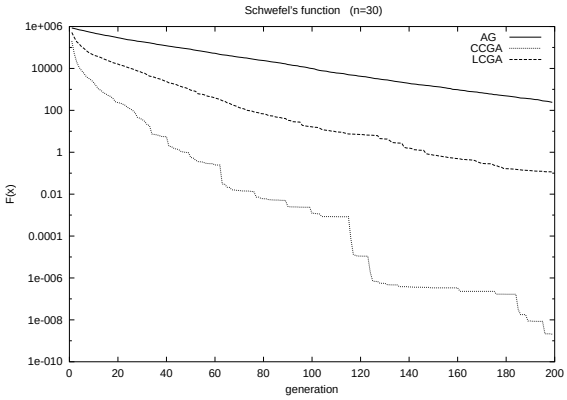


b)

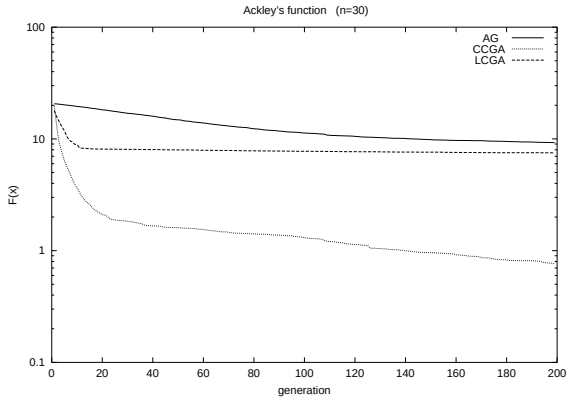


c)

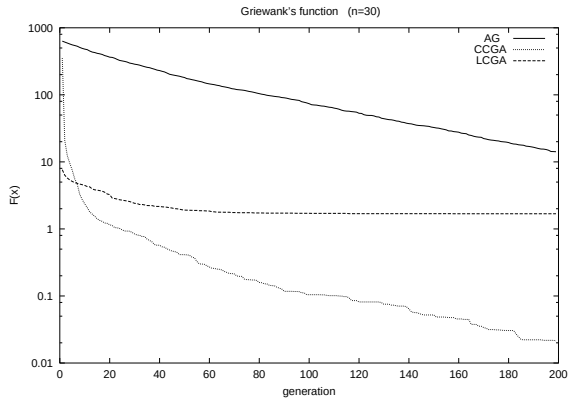
Fig. 1. Sphere model (a), Rosenbrock's function (b) and Rastrigin's function (c)



a)



b)



c)

Fig. 2. Schwefel's function (a), Ackley's function (b) and Griewank's function (c)

sum of local goals. For such unimodal problems (sphere model) LCGA clearly shows its ability of fast (a number of generations) parallel and distributed optimization at low computational cost (no need for communication between agents to collect values of local functions and calculate a value of a global function) and high quality of solution, better than offered by CCGA. LCGA poseses such a ability also for highly multimodal functions (Rastrigin's function) expressed as a sum of local functions. However for problems expressed in more complex way (Schwefel's, Ackley's and Griewank's functions) CCGA with cooperation mechanism and a global coordination shows better performance than LCGA.

References

1. Bouvry, P., F. Arbab, F., Seredynski, F. (2000) Distributed evolutionary optimization, in Manifold: Rosenbrock's function case study, Information Sciences **122**, 141-159
2. Dorigo, M., Maniezzo, V. (1993) Parallel Genetic Algorithms: Introduction and Overview of Current Research, Parallel Genetic Algorithms, Stender, J. (ed.), The Netherlands: IOS Press, 5-42
3. Hillis, W. D. (1992) Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure, Artificial Life II, Langton, C. G. et al. (eds.), Addison-Wesley
4. Husbands, P. (1994) Distributed Coevolutionary Genetic Algorithms for Multi-Criteria and Multi-Constraint Optimization, Evolutionary Computing, Fogarty, T. C. (ed.), LNCS 865, Springer, 150-165
5. Michalewicz, Z. (1996) Genetic Algorithms + Data Structures = Evolution Programs, Springer
6. Paredis, J. (1996) Coevolutionary Life-Time Learning, Parallel Problem Solving from Nature – PPSN IV, Voigt, H. -M. et al. (eds.), LNCS 1141, Springer, 72-80
7. Potter, M. A., De Jong, K. A. (1994) A Cooperative Coevolutionary Approach to Function Optimization, Parallel Problem Solving from Nature – PPSN III, Davidor, Y. et al. (eds.), LNCS 866, Springer
8. Seredynski, F. (1994) Loosely Coupled Distributed Genetic Algorithms, Parallel Problem Solving from Nature – PPSN III, Davidor, Y. et al. (eds.), LNCS 866, Springer
9. Seredynski, F. (1997) Competitive Coevolutionary Multi-Agent Systems: The Application to Mapping and Scheduling Problems, Journal of Parallel and Distributed Computing, **47**, 39-57