Real-Parameter Optimization Using the Mutation Step Co-evolution

Petr Pošík
Faculty of Electrical Engineering, Department of Cybernetics
Czech Technical University in Prague
Technická 2, 166 27 Prague 6
Czech Republic
posik@labe.felk.cvut.cz

Abstract- An evolutionary algorithm for the optimization of a function with real parameters is described in this paper. It uses a cooperative co-evolution to breed and reproduce successful mutation steps. The algorithm described herein is then tested on a suite of 10D and 30D reference optimization problems collected for the Special Session on Real-Parameter Optimization of the IEEE Congress on Evolutionary Computation 2005. The results are of mixed quality (as expected), but reveal several interesting aspects of this simple algorithm.

1 Introduction

The real-parameter optimization is an important issue in many areas of human activities. To name a few applications, we need to optimize a vector of real parameters e.g. when tuning a controller to have the optimal performance, when prescribing a composition for a chemical substance so that it has optimal characteristics, etc.

Many various algorithms were used to solve this task ranging from simple hill-climbing to complex population-based algorithms. This task is a typical application area for evolutionary strategies (ES) [1]. They evolve not only the solution vectors but also parameters of the distribution of their mutation vector. Each individual has its own mutation distribution. In the recent years, an evolutionary strategy with covariance matrix adaptation (CMA-ES) [2] became very popular as a very efficient tool for the real-parameter optimization. A new population is created by several mutations of the current point and after evaluation the offspring population is used to update the current point and the covariance matrix. In this way, successful mutation steps are reproduced.

The algorithm presented here could be described as an evolutionary strategy. However, it does not use any covariance matrix, rather it co-evolves [3] a population of mutation steps that turned out to be successful in the previous generations. This population is a substitute of the probabilistic model (the variances or the covariance matrix) used in ES. The new population is then created by mutating all members of the old population, thus allowing the algorithm to perform the search in several areas of the search space, and not only in one area covered by the gaussian cloud (as in the case of CMA-ES).

The behavior of the algorithm is also somewhat reminiscent of the particle swarm optimization algorithms (PSO, see e.g. [4], pages 379–387). They also adapt the mutation steps (the velocity vectors), but they use information from each individual's history along with the global information. The algorithm described in this paper does not take advantage of the relation of individual particles and their velocity vectors — the mutation steps are not 'owned' by the individuals.

This feature determines also the set of optimization problems this algorithm is well-suited for. The fitness landscape must be of such a form that recently used successful mutation step can be successful even for future mutations (i.e. for other population members situated in different areas of the search space). Such a condition is fulfilled for:

- unimodal function, or
- multimodal function which has the fitness landscapes in the neighborhoods of all the local optima similar to each other.

However, if these assumptions are broken it does not mean that the algorithm is not able to solve the problem at hand. In that case, only the advantage of the cooperation through mutation step sharing vanishes and the strategy behaves very much like a strategy with random mutations (the individual mutation steps are not cooperating, but rather fighting against each other).

The paper is organized as follows. Section 2 introduces the algorithm, in Sec. 3 the results for 10D and 30D test functions are presented, the discussion of results can be found in Sec. 4 and Sec. 5 concludes the paper.

2 Algorithm

The algorithm presented here works with two populations. The first one is the population of potential solutions of generation g, $X^{(g)} = [x_i^{(g)} : i \in \{1, .., N_x\}]$. This population is evaluated by invoking the fitness function on individual population members, $f(x_i^{(g)})$. The second one is a population of mutation steps (i.e. population of mutation vectors) which turned out to be successful in previous generations, $\Delta^{(g)} = [\delta_j^{(g)} : j \in \{1, .., N_\delta\}]$. For this population, the fitness values are stored in an array $f\delta^{(g)} = [f\delta_j^{(g)}]$. The evaluation of mutation steps is explained later.

2.1 Evolutionary Model

The algorithm can be shortly outlined as follows:

1. Initialize both populations
2. Evaluate the population of solutions

3. Create a set of candidate solutions and related candidate mutation steps
4. Evaluate candidate solutions and mutation steps
5. Select better candidate solutions and mutation steps to compete for survival
6. Replace worse population members by selected candidate solutions and mutation steps
7. If the termination criteria are not met, go to 3.

A detailed description of all the algorithm components is presented in the next subsection.

2.2 Algorithm Components

The individual steps used in the above algorithm description are not complex but need to be clarified.

Initialization. The population of solutions, $X^{(0)}$, is initialized randomly in the whole search space. The members of the population of successful mutation steps, $\Delta^{(0)}$, are initialized to zero vectors. All these candidate mutation steps are perturbed before using so that non-zero vectors will emerge during the evolution.

One mutation step using the arithmetic crossover. When generating new candidate solutions, one of them is generated using crossover-like approach in the following way. First, two parents, x_{p1} and x_{p2}, are selected randomly (i.e. with equal probabilities) from the population $X^{(g)}$. A new candidate solution, x_c, is randomly generated on the line connecting the two parents:

$$x'_c = x_{p1} + r(x_{p2} - x_{p1}),$$

where $r \sim U(0,1)$. From both parents only the one closer to the candidate x_c is retained and used for calculation of the resulting mutation step.

$$x'_p = \begin{cases} x_{p1} & \text{if } (r \leq 0.5) \\ x_{p2} & \text{if } (r > 0.5) \end{cases}$$

The resulting mutation step (created by crossover) is

$$\delta'_c = x'_c - x'_p.$$

This operation can introduce useful vectors to the population of mutation steps.

Creating the rest of candidate solutions. To create the major part of the new candidate population, a mutation approach is used. For each of the first $N_x - 1$ population members (i.e. $i \in \{1, .., N_x - 1\}$ now), mutation vector, δ'_i, is randomly selected from the population of successful mutation steps, $\Delta^{(g)}$, and perturbed a bit. (A random vector with the gaussian distribution with diagonal covariance matrix is added to the δ'_i. The diagonal elements of the covariance matrix are a fraction, say 0.0001, of the range that is used by the population of solutions in each of the dimensions.) Then, *two*[1] new candidate solutions, x'_i and x''_i, are created on the opposite sides of the mutated vector, x_i:

$$\begin{aligned} x'_i &= x_i + \delta'_i \\ x''_i &= x_i - \delta'_i \end{aligned}$$

The generated vectors x'_i and x''_i form two populations of candidate vectors; the set of used mutation steps δ'_i forms a population of candidate mutation steps. All these candidate populations have to be evaluated.

Evaluation of candidate populations. The candidate populations (along with the crossover candidate point, x_c) are evaluated via the given fitness function. This gives us the fitness values $f(x'_i)$, $f(x''_i)$, and $f(x_c)$.

The candidate mutation steps are evaluated using their success. The success of the mutation steps can be defined (assuming a minimization problem) as

$$\begin{aligned} f\delta_{+i} &= f(x_i) - f(x'_i), \\ f\delta_{-i} &= f(x_i) - f(x''_i), \\ f\delta_c &= f(x_p) - f(x_c). \end{aligned}$$

Selection of better offspring. Only better of the candidate vectors and mutation steps are selected to compete for survival. Their worse siblings are thrown away.

$$\begin{aligned} \text{if } (f(x'_i) < f(x''_i)) \quad &: \quad f\delta_i = f\delta_{+i} \\ \text{else} \quad &: \quad x'_i = x''_i, \\ &\quad\ \delta'_i = -\delta'_i, \\ &\quad\ f\delta_i = f\delta_{-i} \end{aligned}$$

Furthermore, we add to these offspring populations the candidate created by crossover.

$$X' = X' \cup x_c\,, \quad \Delta' = \Delta' \cup \delta_c$$

From now on, only x'_i's and δ'_i's (along with their fitness values) are considered to be offspring individuals.

Replacement in the population of potential solutions. The offspring candidates x'_i compete for the survival with their respective parents. The only exception is the candidate x_c created by crossover which competes with the last member (unused for mutation) of the parent population $X^{(g)}$ and not with its assigned parent. This feature greatly helps to prevent the premature convergence.

$$\begin{aligned} \text{if } (f(x_i) < f(x'_i)) \quad &: \quad x_i^{(g+1)} = x_i\,, \\ \text{else} \quad &: \quad x_i^{(g+1)} = x'_i\,. \end{aligned}$$

Now, we have the population $X^{(g+1)}$ which passes to the next generation.

Replacement in the population of successful mutation steps. First, fitness values of all the mutation steps in the old population $\Delta^{(g)}$ are reduced by factor s, $s \in \langle 0, 1\rangle$.[2]

$$f\delta'_j = s \cdot f\delta_j^{(g)}$$

[1] The concept of generating two candidate solutions on the opposite sides of the parental vector is important to preserve the mutation strategy unbiased, regardless of the mutation steps population content.

[2] Reducing the fitness of old successful mutation steps guarantees that an early very successful mutation step can be replaced by a less successful, but more recent mutation step. This allows for adaptation of the mutation steps structure.

Then, the fitness of candidate mutation steps is normalized in the following way:

$$f\delta_i = f\delta_i - \min f\delta_i + 1$$

The population of old mutation steps is expanded with the population of candidate mutation steps.

$$\Delta' = \Delta^{(g)} \cup \Delta'$$
$$f\delta = f\delta' \cup f\delta$$

Using truncation selection, the best N_δ mutation steps are selected.

$$\begin{aligned} \text{IndicesOfBest} &= \text{TruncSel}(N_\delta, f\delta) \\ f\delta^{(g+1)} &= f\delta(\text{IndicesOfBest}) \\ \Delta^{(g+1)} &= \Delta'(\text{IndicesOfBest}) \end{aligned}$$

Closing the generation loop. Now we have the population of solution vectors, $X^{(g+1)}$, along with their fitness values, $f(x_i^{(g+1)})$, and a population of successful mutation steps, $\Delta^{(g+1)}$, with their fitness values $f\delta^{(g+1)}$. The generation loop can be closed now by increasing the generation counter, $g = g + 1$.

2.3 Handling Outliers

From the description of the algorithm, it is clear that some of the newly generated points can get out of the bounding hypercube. In the experiments, this bounding hypercube was considered to be a hard bound, so that no points outside were evaluated. For each new individual it is checked which components of the individual vector lie outside the respective feasible interval, and these components are randomly reinitialized to lie inside the hypercube. Only after this correction the individual is evaluated and used in further evolution.

2.4 Algorithm Restarting

During evolution, the algorithm can get to a state when it does not generate any improving candidates. In that case the algorithm is restarted.

Two conditions were used to trigger the reinitialization:

- if the current 'epoch' of the algorithm lasts at least 2000 evaluations and the population has not changed for at least 1000 evaluations (the epoch is a time interval between the last restart of the algorithm and the current time), or
- if the maximal length of the edges of the hypercube in which the population lives drops under 10^{-6}.

Only the population of the solutions is reinitialized, the population of mutation steps is left intact. (The effect of the mutation steps population reinitialization was not studied.)

2.5 Parameters and Settings

The algorithm described above has several parameters which have to be set. The population size is chosen to be greater than the dimension of the search space. The possible outcomes of the arithmetic crossover then lie in the span of the population members. The mutation step population size is then selected to be greater than the size of the solution population, so that it is ensured that some of the old mutation steps always survive to the next generation.

Parameter	**Setting**
Population Size	$N_x = D + 10$
Mut. Step Pop. Size	$N_\delta = N_x + 20$
Mut. Step Fitness Shrinking Factor	$s = 0.9$

The actual setting of individual parameters (including the values used to restart the algorithm) is not based on any theoretical foundations, but rather on an experience with the algorithm behavior.

3 Results

The test functions, monitored statistics and experimental methodology are described on the web page of the special session on Real-Parameter Optimization at IEEE CEC 2005 or in the report [5]. The following subsections present the results for 10D and 30D functions.

3.1 Error Values

Error values of the solutions found by the algorithm are presented in Tables 1, 2, and 3 for 10D, and in Tables 4, 5, and 6 for 30D.

3.2 Function Evaluations Needed

The number of function evaluations needed to achieve the prescribed accuracy level is presented in Tables 7 and 8 for 10D and 30D, respectively. The algorithm did not succeed to find the global optima many times.

	Best		Median		Worst			Success	Success
Problem	1st	7th	13th	19th	25th	Mean	StDev	Rate	Perf.
1	20231	21997	22653	23050	26471	22768	1442	100%	22768.00
2	23694	26092	27204	28783	32199	27372	2078	100%	27371.80
3	37281	40756	43213	46060	51442	43632	3504	100%	43632.00
4	39606	44421	46394	49180	56504	47374	4544	100%	47373.90
5-25								0%	

Table 7: Number of FES needed to achieve the fixed accuracy level for 10D functions.

3.3 Convergence Graphs

Figures 1 to 5 show the evolution of median best-so-far value.

3.4 Algorithm Complexity

All experiments were carried out on a Windows XP machine with Pentium 4 3.4GHz CPU and 1GB of memory running MATLAB 6.5. The statistics intended to describe the complexity of the algorithm are presented in Table 9.

FES	Statistics	Problem								
		1	2	3	4	5	6	7	8	9
1000	1st (Best)	1.74244e+003	1.34176e+003	1.03169e+007	3.59733e+003	6.81810e+003	3.80778e+007	1.09736e+002	2.03080e+001	4.07713e+001
	7th	2.89964e+003	6.91557e+003	2.98089e+007	8.82131e+003	1.12434e+004	3.05664e+008	2.03044e+002	2.05654e+001	5.84886e+001
	13th (Median)	4.58512e+003	1.03995e+004	4.99501e+007	1.27940e+004	1.20589e+004	4.62168e+008	2.88134e+002	2.06321e+001	7.25952e+001
	19th	5.58373e+003	1.38592e+004	6.27097e+007	1.50421e+004	1.39125e+004	6.83332e+008	4.18599e+002	2.07206e+001	7.81214e+001
	25 (Worst)	1.23127e+004	1.96094e+004	8.17758e+007	2.13608e+004	1.65921e+004	1.20924e+009	7.43769e+002	2.08834e+001	1.07150e+002
	Mean	4.63379e+003	1.01453e+004	4.62843e+007	1.24473e+004	1.22236e+004	5.21158e+008	3.38103e+002	2.06296e+001	7.07834e+001
	StDev	2.35143e+003	4.29043e+003	2.15951e+007	4.65688e+003	2.37356e+003	3.22928e+008	1.76834e+002	1.42644e-001	1.62906e+001
10.000	1st (Best)	8.98400e-003	1.00708e+000	3.02111e+004	7.34117e+001	1.37076e+003	2.44546e+002	5.42645e-001	2.01482e+001	1.90382e+001
	7th	4.49639e-002	1.09703e+001	5.15262e+004	2.26523e+002	2.57105e+003	9.03868e+002	6.95722e-001	2.03080e+001	3.01933e+001
	13th (Median)	8.12789e-002	1.91089e+001	7.96698e+004	2.92138e+002	3.14566e+003	1.18951e+003	7.73232e-001	2.04102e+001	3.95902e+001
	19th	1.57424e-001	3.10843e+001	1.61853e+005	4.87544e+002	4.04230e+003	2.17018e+003	8.49267e-001	2.04965e+001	4.82611e+001
	25 (Worst)	7.27320e-001	8.11175e+001	3.83038e+005	1.84173e+003	5.75940e+003	5.64389e+003	1.02289e+000	2.06022e+001	8.22299e+001
	Mean	1.48892e-001	2.30971e+001	1.06814e+005	4.17592e+002	3.37417e+003	1.54209e+003	7.74451e-001	2.03956e+001	4.11505e+001
	StDev	1.75433e-001	1.87258e+001	8.08775e+004	3.76288e+002	1.16148e+003	1.18650e+003	1.18514e-001	1.22411e-001	1.60039e+001
100.000	1st (Best)	4.59556e-009T	5.00478e-009T	5.66757e-009T	6.12619e-009T	9.16523e-002	1.27211e+000	1.01700e-002	2.01391e+001	8.98780e+000
	7th	8.66976e-009T	8.05613e-009T	7.81722e-009T	8.01867e-009T	7.01120e-001	8.05109e+000	2.75527e-002	2.02159e+001	1.30985e+001
	13th (Median)	9.27599e-009T	9.15253e-009T	8.75821e-009T	8.66862e-009T	8.61175e-001	9.67686e+000	3.62197e-002	2.02847e+001	1.93937e+001
	19th	9.65395e-009T	9.51059e-009T	9.46301e-009T	9.44232e-009T	1.90969e+000	1.40084e+001	5.16803e-002	2.03386e+001	2.49247e+001
	25 (Worst)	9.93379e-009T	9.99557e-009T	9.99751e-009T	9.90025e-009T	1.64876e+001	4.07603e+001	5.94898e-002	2.04092e+001	3.09298e+001
	Mean	8.83358e-009T	8.59865e-009T	8.48558e-009T	8.54705e-009T	2.13319e+000	1.24633e+001	3.70507e-002	2.02745e+001	1.91946e+001
	StDev	1.33317e-009T	1.36648e-009T	1.34518e-009T	9.85134e-010T	3.57362e+000	8.97498e+000	1.51060e-002	7.65884e-002	6.61912e+000

Table 1: Error values for 10D functions nr. 1 – 9

FES	Statistics	Problem							
		10	11	12	13	14	15	16	17
1000	1st (Best)	5.80184e+001	8.12639e+000	2.34376e+004	6.30406e+000	3.66299e+000	3.48160e+002	2.58540e+002	3.67184e+002
	7th	8.34984e+001	1.08348e+001	3.70031e+004	8.47696e+000	4.20523e+000	6.23096e+002	3.79627e+002	4.18935e+002
	13th (Median)	9.12955e+001	1.14000e+001	5.19865e+004	9.55827e+000	4.29225e+000	6.65287e+002	4.05589e+002	4.52428e+002
	19th	1.05037e+002	1.21489e+001	6.40195e+004	1.12953e+001	4.41822e+000	7.46938e+002	4.45467e+002	5.34589e+002
	25 (Worst)	1.39114e+002	1.29686e+001	1.11289e+005	1.41311e+001	4.57787e+000	8.94522e+002	5.00999e+002	5.89762e+002
	Mean	9.37138e+001	1.12563e+001	5.39104e+004	1.00038e+001	4.28996e+000	6.74991e+002	4.07724e+002	4.67231e+002
	StDev	2.08454e+001	1.19908e+000	2.29027e+004	2.11055e+000	2.04154e-001	1.13250e+002	5.96920e+001	6.57991e+001
10.000	1st (Best)	1.84312e+001	7.05768e+000	7.88701e+002	1.13395e+000	3.55806e+000	2.02044e+002	1.53082e+002	2.40266e+002
	7th	3.10792e+001	9.48647e+000	2.37503e+003	1.80171e+000	3.88298e+000	3.37415e+002	1.82735e+002	2.89874e+002
	13th (Median)	4.03056e+001	1.03768e+001	4.63494e+003	2.15159e+000	4.07143e+000	3.96404e+002	2.44913e+002	2.98743e+002
	19th	5.01230e+001	1.14371e+001	6.42196e+003	2.64539e+000	4.14020e+000	4.91121e+002	2.58992e+002	3.38652e+002
	25 (Worst)	7.33024e+001	1.22901e+001	1.50636e+004	3.66858e+000	4.29082e+000	6.40715e+002	3.03331e+002	4.08258e+002
	Mean	4.13712e+001	1.01611e+001	5.03365e+003	2.22386e+000	4.01066e+000	4.21837e+002	2.34009e+002	3.12355e+002
	StDev	1.42797e+001	1.38933e+000	3.49392e+003	6.39870e-001	2.02222e-001	1.14965e+002	4.76666e+001	4.17586e+001
100.000	1st (Best)	1.49815e+001	6.78285e+000	2.66206e+001	4.68998e-001	3.26001e+000	1.37254e+002	1.23367e+002	1.45125e+002
	7th	1.99161e+001	8.47390e+000	9.46710e+001	7.88741e-001	3.59021e+000	2.22634e+002	1.55445e+002	1.92432e+002
	13th (Median)	2.68732e+001	9.29237e+000	1.20325e+002	1.03498e+000	3.70433e+000	2.69885e+002	1.74730e+002	2.07081e+002
	19th	3.18432e+001	9.65436e+000	2.05850e+002	1.37700e+000	3.82959e+000	3.34064e+002	1.86711e+002	2.32234e+002
	25 (Worst)	4.97850e+001	1.04512e+001	3.73240e+003	2.14617e+000	4.13566e+000	5.56476e+002	2.73042e+002	2.97348e+002
	Mean	2.67652e+001	9.02888e+000	6.04531e+002	1.13652e+000	3.70647e+000	2.93771e+002	1.77178e+002	2.11816e+002
	StDev	8.64571e+000	1.02016e+000	1.07364e+003	4.33100e-001	2.30358e-001	1.02074e+002	3.15727e+001	3.37456e+001

Table 2: Error values for 10D functions nr. 10 – 17

FES	Statistics	Problem							
		18	19	20	21	22	23	24	25
1000	1st (Best)	1.03254e+003	1.08809e+003	1.08326e+003	1.08529e+003	1.06926e+003	1.17765e+003	8.79258e+002	1.10784e+003
	7th	1.14329e+003	1.14224e+003	1.13777e+003	1.31648e+003	1.13850e+003	1.32706e+003	1.24639e+003	1.26327e+003
	13th (Median)	1.16573e+003	1.17500e+003	1.16325e+003	1.35489e+003	1.19282e+003	1.35561e+003	1.31553e+003	1.30950e+003
	19th	1.20930e+003	1.21273e+003	1.18964e+003	1.38227e+003	1.26354e+003	1.37411e+003	1.34789e+003	1.35509e+003
	25 (Worst)	1.32844e+003	1.32021e+003	1.24930e+003	1.43718e+003	1.35507e+003	1.47321e+003	1.42444e+003	1.38206e+003
	Mean	1.17403e+003	1.18306e+003	1.16577e+003	1.33887e+003	1.20158e+003	1.34369e+003	1.27518e+003	1.30033e+003
	StDev	5.80399e+001	6.38609e+001	4.40212e+001	7.02220e+001	7.88999e+001	6.12655e+001	1.31362e+002	7.62397e+001
10.000	1st (Best)	8.38080e+002	6.86832e+002	6.74195e+002	6.18560e+002	7.89366e+002	7.76724e+002	4.48385e+002	2.08233e+002
	7th	9.98985e+002	8.82196e+002	9.57563e+002	8.60739e+002	8.29700e+002	1.10894e+003	6.51169e+002	3.30106e+002
	13th (Median)	1.02894e+003	1.01897e+003	1.01443e+003	1.05839e+003	8.57222e+002	1.18332e+003	1.01513e+003	8.09027e+002
	19th	1.06171e+003	1.06401e+003	1.06869e+003	1.16840e+003	9.14158e+002	1.24657e+003	1.08516e+003	1.19094e+003
	25 (Worst)	1.12350e+003	1.09100e+003	1.09636e+003	1.24093e+003	1.03431e+003	1.27784e+003	1.22871e+003	1.30760e+003
	Mean	1.02498e+003	9.74082e+002	9.83288e+002	1.00131e+003	8.74189e+002	1.14814e+003	9.03192e+002	7.90339e+002
	StDev	5.88449e+001	1.08049e+002	1.18424e+002	1.97355e+002	5.76192e+001	1.29841e+002	2.58744e+002	4.10564e+002
100.000	1st (Best)	8.00057e+002	5.00022e+002	5.00207e+002	2.00073e+002	3.00451e+002	4.25173e+002	2.00001e+002	2.00000e+002
	7th	8.07439e+002	8.00360e+002	8.02507e+002	4.13784e+002	7.88431e+002	7.42505e+002	2.00036e+002	2.00003e+002
	13th (Median)	9.00911e+002	8.08026e+002	8.61724e+002	5.00342e+002	7.98964e+002	8.23041e+002	2.01404e+002	2.00020e+002
	19th	9.75283e+002	9.32117e+002	9.91551e+002	9.43121e+002	8.04520e+002	9.70775e+002	2.92000e+002	2.01237e+002
	25 (Worst)	1.03393e+003	1.02457e+003	1.03935e+003	1.13064e+003	8.45619e+002	1.09476e+003	1.09026e+003	8.49007e+002
	Mean	9.01543e+002	8.44501e+002	8.62896e+002	6.34934e+002	7.78859e+002	8.34559e+002	3.13835e+002	2.57319e+002
	StDev	8.70504e+001	1.33741e+002	1.47336e+002	3.00400e+002	1.01998e+002	1.64184e+002	2.19246e+002	1.55310e+002

Table 3: Error values for 10D functions nr. 18 – 25

FES	Statistics	Problem								
		1	2	3	4	5	6	7	8	9
1000	1st (Best)	5.33585e+004	6.16494e+004	5.56986e+008	7.44209e+004	3.14520e+004	1.31495e+010	2.78878e+003	2.10999e+001	3.59050e+002
	7th	5.94619e+004	7.66128e+004	9.72597e+008	1.02520e+005	3.63415e+004	3.14673e+010	3.85054e+003	2.11502e+001	4.02055e+002
	13th (Median)	6.64496e+004	9.14726e+004	1.07867e+009	1.23692e+005	3.84696e+004	3.88747e+010	4.64674e+003	2.12153e+001	4.34512e+002
	19th	7.28793e+004	1.06593e+005	1.44014e+009	1.39580e+005	3.95729e+004	4.15270e+010	5.57934e+003	2.12350e+001	4.60435e+002
	25 (Worst)	8.59822e+004	1.43638e+005	1.89233e+009	1.97962e+005	4.39971e+004	7.28580e+010	7.85845e+003	2.12613e+001	5.07010e+002
	Mean	6.68271e+004	9.26420e+004	1.19472e+009	1.22887e+005	3.77665e+004	3.81332e+010	4.80270e+003	2.11926e+001	4.31161e+002
	StDev	8.85247e+003	2.04798e+004	3.38759e+008	2.87738e+004	3.16733e+003	1.25593e+010	1.37373e+003	5.36316e-002	4.04042e+001
10.000	1st (Best)	4.83026e+003	2.41472e+004	5.02534e+007	3.34219e+004	1.87195e+004	2.59226e+008	3.09827e+002	2.09694e+001	2.53693e+002
	7th	6.26610e+003	2.92015e+004	9.09276e+007	4.40980e+004	2.29835e+004	8.58493e+008	4.68987e+002	2.09984e+001	2.82094e+002
	13th (Median)	7.50153e+003	3.29036e+004	1.17449e+008	5.21934e+004	2.46946e+004	1.05542e+009	5.40753e+002	2.10489e+001	3.01453e+002
	19th	8.91280e+003	3.48984e+004	1.45926e+008	5.97914e+004	2.62733e+004	1.51664e+009	6.64287e+002	2.10869e+001	3.09794e+002
	25 (Worst)	1.27283e+004	4.74157e+004	2.28189e+008	7.20726e+004	3.13359e+004	2.37650e+009	8.95632e+002	2.11261e+001	3.27128e+002
	Mean	7.78853e+003	3.27281e+004	1.24130e+008	5.18018e+004	2.45469e+004	1.18207e+009	5.70517e+002	2.10496e+001	2.96272e+002
	StDev	2.03784e+003	5.49452e+003	4.56759e+007	1.02363e+004	2.93008e+003	5.42098e+008	1.45480e+002	5.01784e-002	1.92178e+001
100.000	1st (Best)	3.81564e-003	1.09078e+002	4.78335e+005	9.84148e+003	1.31734e+004	1.76593e+003	1.09451e+000	2.07860e+001	9.75062e+001
	7th	8.46735e-002	2.34731e+002	1.05326e+006	1.66972e+004	1.42792e+004	3.96595e+003	1.38578e+000	2.09501e+001	1.32598e+002
	13th (Median)	3.43043e-001	5.12535e+002	1.24714e+006	1.73931e+004	1.51711e+004	9.71290e+003	2.03212e+000	2.09776e+001	1.39917e+002
	19th	1.26075e+000	7.17709e+002	2.26181e+006	2.15464e+004	1.58729e+004	3.00221e+004	3.66481e+000	2.09907e+001	1.67014e+002
	25 (Worst)	3.41721e+001	1.51406e+003	3.98283e+006	3.51225e+004	1.92038e+004	8.40980e+004	7.65190e+000	2.10482e+001	2.89599e+002
	Mean	2.59086e+000	5.50265e+002	1.73851e+006	1.93115e+004	1.51696e+004	1.90744e+004	2.70803e+000	2.09653e+001	1.52427e+002
	StDev	7.12183e+000	3.70340e+002	9.70251e+005	5.19592e+003	1.32832e+003	2.20840e+004	1.78370e+000	5.43384e-002	3.98847e+001
300.000	1st (Best)	8.29107e-009T	8.80385e-009T	2.59134e-003	8.39365e+002	6.93788e+003	7.71753e+001	7.36657e-007	2.07233e+001	9.16293e+001
	7th	4.78684e-003	5.13011e-007	2.27437e-001	2.53832e+003	7.54526e+003	1.54362e+002	1.01986e-003	2.08895e+001	1.13906e+002
	13th (Median)	1.22952e-002	6.94963e-004	3.05088e+000	4.39145e+003	8.12566e+003	3.13523e+002	1.06204e-002	2.09109e+001	1.33577e+002
	19th	9.35114e-002	4.23037e-002	1.45479e+002	5.31771e+003	9.39549e+003	1.28986e+003	3.77824e-002	2.09398e+001	1.39457e+002
	25 (Worst)	1.17818e+001	7.56331e+000	6.72351e+003	1.26939e+004	9.94298e+003	6.14261e+003	9.89521e-001	2.09976e+001	1.83245e+002
	Mean	7.96501e-001	4.40181e-001	3.67138e+002	4.79776e+003	8.34362e+003	1.21108e+003	1.41199e-001	2.09003e+001	1.31352e+002
	StDev	2.49062e+000	1.52079e+000	1.34529e+003	3.43707e+003	1.04132e+003	1.83401e+003	2.83291e-001	6.82016e-002	2.38778e+001

Table 4: Error values for 30D functions nr. 1 – 9

FES	Statistics	Problem							
		10	11	12	13	14	15	16	17
1000	1st (Best)	5.75127e+002	4.10244e+001	1.23489e+006	9.68059e+001	1.38274e+001	8.85506e+002	7.57884e+002	8.19386e+002
	7th	6.32923e+002	4.40339e+001	1.53357e+006	1.89401e+002	1.40558e+001	1.09102e+003	9.16665e+002	1.00666e+003
	13th (Median)	6.77973e+002	4.49563e+001	1.63304e+006	2.64859e+002	1.41140e+001	1.13270e+003	9.53444e+002	1.10126e+003
	19th	7.25612e+002	4.57573e+001	1.68616e+006	3.22076e+002	1.42168e+001	1.16385e+003	1.05617e+003	1.20258e+003
	25 (Worst)	8.27592e+002	4.68403e+001	1.98498e+006	4.29839e+002	1.44380e+001	1.22201e+003	1.15200e+003	1.45879e+003
	Mean	6.84078e+002	4.47948e+001	1.61242e+006	2.57756e+002	1.41079e+001	1.11831e+003	9.74924e+002	1.10242e+003
	StDev	6.27582e+001	1.46126e+000	1.82697e+005	8.99537e+001	1.53017e-001	6.77495e+001	9.71689e+001	1.42540e+002
10.000	1st (Best)	3.60825e+002	3.62692e+001	2.28281e+005	2.72943e+001	1.34347e+001	5.71730e+002	4.57131e+002	4.29601e+002
	7th	4.27560e+002	3.86685e+001	3.38787e+005	2.97526e+001	1.36711e+001	6.45172e+002	5.39936e+002	6.48339e+002
	13th (Median)	4.42273e+002	3.97414e+001	3.82449e+005	3.16659e+001	1.37825e+001	6.82604e+002	5.63908e+002	7.11482e+002
	19th	4.51072e+002	4.15162e+001	4.37251e+005	3.30554e+001	1.39168e+001	7.44512e+002	5.88142e+002	7.53161e+002
	25 (Worst)	4.78391e+002	4.38430e+001	6.26811e+005	3.85523e+001	1.41314e+001	8.16676e+002	6.46716e+002	8.48293e+002
	Mean	4.36965e+002	4.01154e+001	3.92326e+005	3.20681e+001	1.37979e+001	6.95428e+002	5.61405e+002	6.95798e+002
	StDev	2.94274e+001	1.89426e+000	8.37716e+004	3.00627e+000	1.76060e-001	6.45442e+001	4.02557e+001	9.91744e+001
100.000	1st (Best)	1.70538e+002	3.41254e+001	7.03442e+004	6.73160e+000	1.28458e+001	3.02807e+002	2.85371e+002	3.73665e+002
	7th	2.19633e+002	3.77702e+001	1.00889e+005	9.47320e+000	1.35499e+001	4.55661e+002	3.62355e+002	5.04463e+002
	13th (Median)	2.47865e+002	3.82237e+001	2.26197e+005	1.08902e+001	1.36589e+001	5.12774e+002	4.08973e+002	5.35252e+002
	19th	2.73380e+002	3.95855e+001	2.57257e+005	1.19978e+001	1.36868e+001	5.58757e+002	4.56105e+002	5.59479e+002
	25 (Worst)	3.02847e+002	4.28446e+001	3.23785e+005	1.82203e+001	1.39086e+001	6.31450e+002	5.24023e+002	6.62797e+002
	Mean	2.44953e+002	3.87279e+001	2.02033e+005	1.10128e+001	1.36055e+001	4.97591e+002	4.05638e+002	5.30783e+002
	StDev	3.47914e+001	1.96431e+000	8.09392e+004	2.71396e+000	2.00245e-001	8.88457e+001	6.40851e+001	6.41770e+001
300.000	1st (Best)	1.69471e+002	3.41254e+001	1.34641e+004	5.19963e+000	1.24592e+001	2.96856e+002	2.76942e+002	3.03003e+002
	7th	2.01319e+002	3.72753e+001	2.25021e+004	7.63531e+000	1.31763e+001	4.03419e+002	3.60970e+002	4.33356e+002
	13th (Median)	2.35159e+002	3.77020e+001	8.52881e+004	8.89197e+000	1.33247e+001	4.14009e+002	3.87778e+002	4.64531e+002
	19th	2.56069e+002	3.84652e+001	1.73916e+005	1.02407e+001	1.34390e+001	4.31006e+002	4.13802e+002	4.76037e+002
	25 (Worst)	2.91180e+002	4.16462e+001	2.19283e+005	1.68078e+001	1.36077e+001	4.76033e+002	4.43476e+002	5.53928e+002
	Mean	2.32350e+002	3.77005e+001	1.01431e+005	9.01956e+000	1.32465e+001	4.10633e+002	3.81230e+002	4.54394e+002
	StDev	3.50595e+001	1.52712e+000	7.22294e+004	2.27836e+000	2.92312e-001	4.36705e+001	4.53817e+001	5.65009e+001

Table 5: Error values for 30D functions nr. 10 – 17

FES	Statistics	Problem							
		18	19	20	21	22	23	24	25
1000	1st (Best)	1.23666e+003	1.17935e+003	1.16585e+003	1.35494e+003	1.46455e+003	1.37025e+003	1.42107e+003	1.47573e+003
	7th	1.29718e+003	1.27221e+003	1.30709e+003	1.40953e+003	1.55119e+003	1.40338e+003	1.45353e+003	1.52806e+003
	13th (Median)	1.32700e+003	1.31885e+003	1.33340e+003	1.43067e+003	1.60022e+003	1.42717e+003	1.48809e+003	1.54105e+003
	19th	1.36503e+003	1.34985e+003	1.36576e+003	1.45654e+003	1.75308e+003	1.45047e+003	1.51988e+003	1.61049e+003
	25 (Worst)	1.41423e+003	1.39187e+003	1.40812e+003	1.52649e+003	1.80164e+003	1.50213e+003	1.53330e+003	1.67815e+003
	Mean	1.32547e+003	1.30861e+003	1.32514e+003	1.43303e+003	1.63129e+003	1.43004e+003	1.48536e+003	1.56600e+003
	StDev	4.50637e+001	5.20935e+001	5.59364e+001	4.38720e+001	1.08508e+002	3.36861e+001	3.45352e+001	5.68263e+001
10.000	1st (Best)	1.09302e+003	1.08503e+003	1.05010e+003	1.20237e+003	1.16384e+003	1.18491e+003	1.26737e+003	1.22469e+003
	7th	1.14978e+003	1.12820e+003	1.13376e+003	1.23593e+003	1.22206e+003	1.24218e+003	1.30263e+003	1.27037e+003
	13th (Median)	1.15825e+003	1.15120e+003	1.15738e+003	1.24344e+003	1.26344e+003	1.26125e+003	1.33074e+003	1.29400e+003
	19th	1.17776e+003	1.17856e+003	1.18385e+003	1.25279e+003	1.28528e+003	1.27004e+003	1.34454e+003	1.31860e+003
	25 (Worst)	1.20113e+003	1.23324e+003	1.26046e+003	1.27252e+003	1.35309e+003	1.30805e+003	1.36912e+003	1.33465e+003
	Mean	1.15867e+003	1.14917e+003	1.15727e+003	1.24277e+003	1.25368e+003	1.25524e+003	1.32347e+003	1.29050e+003
	StDev	2.82241e+001	3.59399e+001	4.06148e+001	1.63803e+001	5.15932e+001	2.44811e+001	2.94555e+001	3.10040e+001
100.000	1st (Best)	1.05314e+003	1.04688e+003	1.00394e+003	7.09488e+002	1.10225e+003	7.61800e+002	1.12689e+003	1.01888e+003
	7th	1.09419e+003	1.08179e+003	1.08376e+003	8.51555e+002	1.15685e+003	1.05909e+003	1.18562e+003	1.19843e+003
	13th (Median)	1.11078e+003	1.10311e+003	1.11391e+003	1.02293e+003	1.17449e+003	1.14070e+003	1.21227e+003	1.22657e+003
	19th	1.13852e+003	1.12933e+003	1.13505e+003	1.13363e+003	1.23031e+003	1.19153e+003	1.27347e+003	1.25996e+003
	25 (Worst)	1.18570e+003	1.15382e+003	1.17897e+003	1.21335e+003	1.27393e+003	1.22625e+003	1.33142e+003	1.28621e+003
	Mean	1.11395e+003	1.10428e+003	1.10769e+003	9.93501e+002	1.18482e+003	1.10113e+003	1.22603e+003	1.21726e+003
	StDev	2.99895e+001	3.28115e+001	4.03304e+001	1.54068e+002	4.62375e+001	1.24592e+002	6.04944e+001	6.08417e+001
300.000	1st (Best)	1.01124e+003	1.00806e+003	9.82622e+002	5.00007e+002	1.10225e+003	6.07431e+002	7.60284e+002	5.41279e+002
	7th	1.04077e+003	1.02565e+003	1.02942e+003	5.00041e+002	1.13801e+003	8.10644e+002	1.01706e+003	9.31473e+002
	13th (Median)	1.05927e+003	1.03899e+003	1.05435e+003	5.01399e+002	1.15571e+003	8.58275e+002	1.11909e+003	1.06670e+003
	19th	1.07456e+003	1.07309e+003	1.08253e+003	5.13591e+002	1.16777e+003	1.06339e+003	1.19072e+003	1.17221e+003
	25 (Worst)	1.12623e+003	1.11415e+003	1.15573e+003	1.17799e+003	1.23031e+003	1.21690e+003	1.26739e+003	1.23334e+003
	Mean	1.06085e+003	1.04894e+003	1.05874e+003	6.03521e+002	1.15535e+003	9.21673e+002	1.09659e+003	1.02670e+003
	StDev	3.02455e+001	2.97697e+001	4.32641e+001	2.18080e+002	2.94102e+001	1.81187e+002	1.20763e+002	1.91553e+002

Table 6: Error values for 30D functions nr. 18 – 25

	Best		Median		Worst			Success	Success
Problem	1st	7th	13th	19th	25th	Mean	StDev	Rate	Perf.
1	144551					283820	45081	12%	1376410.00
2	231593	297819				289683	21070	32%	836747.00
3-6								0%	
7	201810	258713				278210	33098	44%	569264.00
8-25								0%	

Table 8: Number of FES needed to achieve the fixed accuracy level for 30D functions.

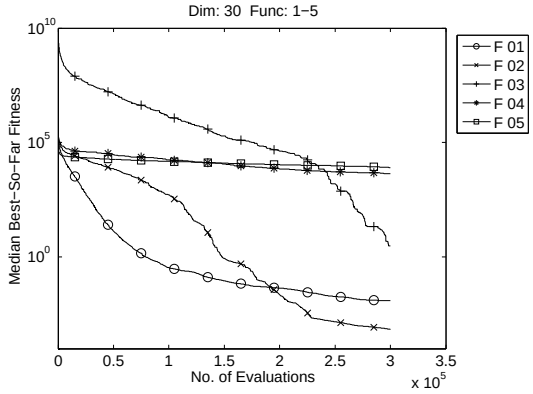

Figure 1: Convergence graph for 30D functions nr. 1 – 5

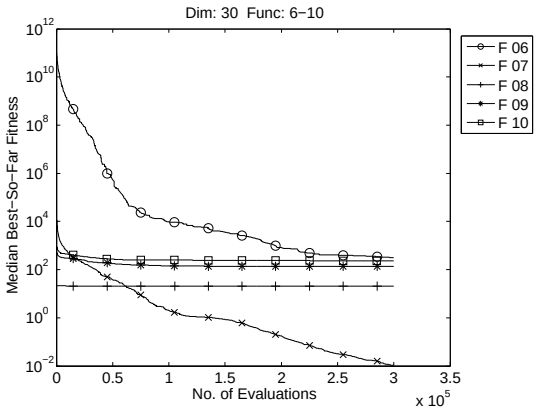

Figure 2: Convergence graph for 30D functions nr. 6 – 10

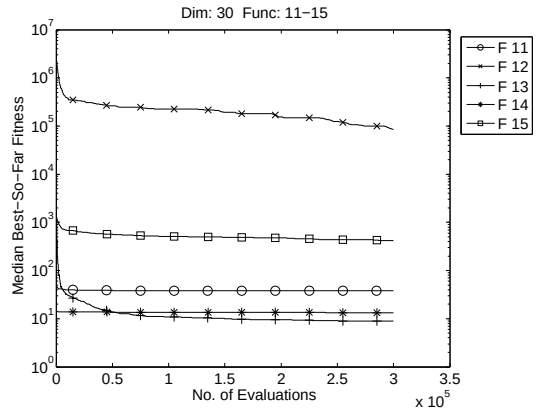

Figure 3: Convergence graph for 30D functions nr. 11 – 15

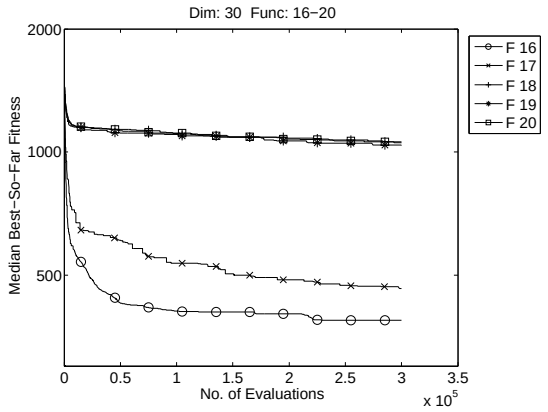

Figure 4: Convergence graph for 30D functions nr. 16 – 20

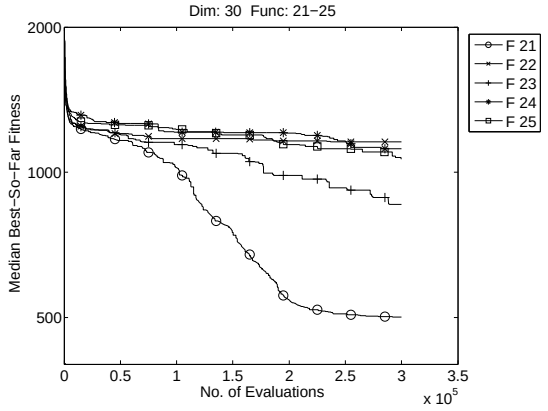

Figure 5: Convergence graph for 30D functions nr. 21 – 25

Dim	T0	T1	$\hat{T}2$	($\hat{T}2$-T1)/T0
10		2.0	22.1	15.5
30	1.3	1.6	16.8	11.7
50		1.7	17.0	11.8

Table 9: Measured times and algorithm complexity

The results presented in Tab. 9 may seem a bit strange (the complexity drops with increasing dimensionality?), but it can be explained by the MATLAB matrix functions. It is much faster to evaluate a matrix with 30 rows than to evaluate a one-row matrix 30 times. Thus, the evolution of 30 dimensional population of size 40 is faster than the evolution of 10 dimensional population of size 20, because the evaluation (and population manipulation) functions are executed less often.

4 Discussion and Improvement Suggestions

In this section, a few observations about the algorithm behavior when solving the benchmark functions are made.

- The algorithm is able to handle problems with *different condition numbers*. As can be seen from Tables 1 and 7, the algorithm solved the 10D versions of problems 1 – 3, but failed to solve the 30D versions (with a few exceptions). This suggests that the perturbation of the mutation steps alone is not able to adapt them successfully in case of high-dimensional functions. Introduction of a factor for mutation step shortening (or extension) along with an adaptation scheme (based e.g. on the 1/5 rule) could greatly improve the results.

- The influence of the *noise in fitness* can be observed on pairs of functions 2–4 and 16–17. In the case of 10D functions, the difference is minor, however, in case of 30D, the algorithm succeeded only for problem 2 and the difference in behavior is substantial (see Fig. 1); in case of the pair 16–17 the algorithm did not find the global optimum. Thus, it is rather worthless to say that the noise caused only slight deterioration of solution quality and that the algorithm behaved similarly in both cases.

- The algorithm should be *rotationally invariant*. The only component that depends on the rotation is the procedure for handling the outliers. The differences that can be seen in pairs 9–10 and 15–16 can be accounted to that procedure.

- The influence of *non-continuity of the fitness* can be observed on the pair 21–23 in Fig. 5. It slightly worsens the results but the changes seem to be statistically insignificant.

- The effect of placing *the global optimum on bounds* is questionable. There is not much difference between the pair 18–20, see Fig. 4 (but the algorithm completely missed the optimum). Based on a relatively

poor performance on function 5, it can be stated that the algorithm would not solve such problems very well. This issue could probably be solved by using a different procedure for handling outliers, namely placing the outliers to the nearest boundaries (or to consider the bounding hypercube to be only a soft bound and allow the algorithm to evaluate the points outside of it). The effect of such a modification must be studied yet.

- Similarly, the influence of *narrow global optimum basin* cannot be described, the algorithm did not found it in either case.
- From the results and from the algorithm description it can be stated that the fact that *the initialization interval does not contain the global optimum* is not an issue for this algorithm. It is able to 'enlarge' the search area.

5 Conclusions

A very simple co-evolutionary approach for real-valued optimization was presented in this paper. The algorithm evolves two populations, one with the potential solutions to the given problem, and the other with mutation steps that proved to be successful in recent generations.

The algorithm has a pleasant behavior on the first unimodal functions, as expected. It can solve even high-conditioned problems thanks to the sharing of successful mutation steps. For the other problems, this feature is not of much benefit, so that the algorithm mainly exploits the local neighborhoods of the population members.

As a promising way of the algorithm improvement, the adaptation of the overall control parameter affecting the mutation step length is suggested. The perturbation of the population of mutation steps as presented here does not seem to be sufficient. In the evolution of the mutation step population, the crossover could be used which might also improve the results.

The comparison of this really simple algorithm with other, perhaps more complex and smart, optimization algorithms which take part in this special session will be very interesting and enriching.

Acknowledgments

The research of Petr Pošík was supported by the research program No. MSM6840770012 "Transdisciplinary Research in the Area of Biomedical Engineering II" of the CTU in Prague, sponsored by the Ministry of Education, Youth and Sports of the Czech Republic.

Bibliography

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., *Evolutionary Computation 1*. IOP Publishing, 2000.

[2] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[3] J. Paredis, "Coevolutionary computation," *Artificial Life*, vol. 2, no. 4, pp. 355–375, 1995.

[4] D. Corne, M. Dorigo, and F. Glover, eds., *New Ideas in Optimization*. McGraw-Hill, 1999.

[5] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 Special Session on Real-Parameter Optimization," tech. rep., Nanyang Technological University, Singapore, May 2005. http://www.ntu.edu.sg/home/epnsugan.