Mean-Adaptive Real-Coding Genetic Algorithm and its Applications to Electromagnetic Optimization (Part One)

Viktor OTEVŘEL, Zbyněk RAIDA

Dept. of Radio Electronics, Brno University of Technology, Purkyňova 118, 612 00 Brno, Czech Republic

viktor.otevrel@seznam.cz, raida@feec.vutbr.cz

Abstract. *In the paper, a novel instance of the real-coding steady-state genetic algorithm, called the Mean-adaptive real-coding genetic algorithm, is put forward. In this instance, three novel implementations of evolution operators are incorporated. Those are a recombination and two mutation operators. All of the evolution operators are designed with the aim of possessing a big explorative power. Moreover, one of the mutation operators exhibits self-adaptive behavior and the other exhibits adaptive behavior, thereby allowing the algorithm to self-control its own mutability as the search advances. This algorithm also takes advantage of population-elitist selection, acting as a replacement policy, being adopted from evolution strategies.*

The purpose of this paper (i.e., the first part) is to provide theoretical foundations of a robust and advanced instance of the real-coding genetic algorithm having the big potential of being successfully applied to electromagnetic optimization.

Keywords

Real-coding genetic optimization, Mean-adaptive real-coding genetic algorithm, mean-adaptive mutation, Gaussian mutation with adaptive step size control, uniform-wise crossover, population-elitist selection.

1. Introduction

The high flexibility and popularity of genetic algorithms (GAs) represent the main factors being responsible for their widespread exploitation within the electromagnetic community. The character of problems developing in electromagnetics is usually too complicated to allow them to be solved analytically. In such cases, numerical models have to be made. These models often represent high-dimensional ill-behaved functions depending upon a mixture of both continuous and discrete variables. In most cases and without having a substantial effect on performance, the discrete variables can simply be treated by the algorithms as if they were defined in continuous search space, and appropriately adjusted prior to their using in evaluation. Hence, in order to fully exploit the domain knowledge, the usage of real-coding GAs (RCGAs) is advisable. The purpose of this paper is as follows:

- To present a robust RCGA solution called the *Mean-adaptive RCGA* (MAD-RCGA).
- To provide an advanced and well-proved tool for single-objective optimization.
- To ease off the deficiency existing in the contemporary electromagnetic-related literature as regards real-coding genetic optimization.

The bases of this algorithm were originally published in [8] where the algorithm was successfully applied to the wide-band optimization of a shielded microstrip line. Since then, the algorithm has been subjected to a lot of tests on standard benchmark functions and real-world problems (e.g., see [7]) and, based on the obtained results, modified into the current shape that will be presented in this paper.

2. MAD-RCGA in Outline

Much of attention in the dedicated literature is paid to examining and exploiting GAs that comprise the following types of replacement [5], [6], [11]:

- Generational – the parental population is replaced by a new population at each generation.
- "Traditional" steady-state – some offspring solutions (usually one or two) are produced at each generation to update the population of parents.

As well-known, instances of the steady-state GA (SSGA) represent good candidates for application to static optimization ([5], [6], [9], [11]) (i.e., non-time-varying environments) or quasi-static optimization (i.e., slowly time-varying environments). These types of optimization constitute the majority of technical problems arising in electromagnetic optimization and hence this chapter is devoted to the study of one of the SSGA instances – MAD-RCGA – that was devised to aid in addressing hard-to-solve technical optimization problems.

Whereas the bulk of SSGA instances take advantage of "traditional" replacement mechanisms, the design of

MAD-RCGA tries to follow another way by incorporating population-elitist selection (PES) [1], [9], [11] into its structure. The reasons for doing so are as follows:

- PES pushes the population with sufficiently strong force towards its convergence (i.e., its collapsing) because of a high level of selection pressure that is in direct proportion to the ratio of λ/μ (i.e., the number of offspring created at each generation to the size of the parent population).
- The dependency of the strength of selection pressure on λ/μ allows the problem-dependent adjustment of both the duration and the scope of the exploration phase. Moreover, this adjustment can also be made dynamically over the course of a run if it is found profitable in search for the optimum.
- Owning to a high level of selection pressure produced by PES, an SSGA instance with this type of replacement can be equipped with variation operators (i.e., recombination or/and mutation) having large exploration scopes (mutation can possibly be working at high rates).
- The usage of PES in combination with highly explorative variation operators has been proved by evolution strategies (ESs) ([1], [9]) to perform well.
- Despite its favorable properties in terms of selection pressure, PES is occasionally exploited in SSGAs in comparison with ESs.

The original intention behind launching the development of MAD-RCGA was to offer a powerful widely applicable optimization tool, inherently working with continuous variables, for addressing complex problems that frequently arise in the domain of electromagnetics. To that end, the structure of this RCGA scheme was designed to take the following form:

- A general parent selection technique can be considered and incorporated into the structure of the algorithm.
- The individual's chromosome is composed of object variables and one strategy parameter associated with the configuration of the algorithm.
- In the place of recombination, a novel crossover operator termed ***u**niform-**w**ise crossover* (UWX) is employed.
- There are two mutation mechanisms, originally proposed in this paper, that are used for mutating. Those are:
  - ***M**ean-**a**daptive **m**utation* (MAM)
  - ***G**aussian **m**utation with the **a**daptive **s**tep **s**ize **c**ontrol based upon the **v**iability of produced **m**utants* (GMASS-CVM)
- As a replacement mechanism, PES is exploited.

Before delving into details, we will at first outline the fundamental principles of the three originally proposed operators – UWX, MAM and GMASS-CVM – in order to provide a basic notion of their functioning prior to describing the related evolutionary scheme. Their working principles can briefly be characterized as follows:

- UWX: It represents a highly explorative multi-parent variable-wise recombination operator working on a principle similar to traditional real-valued *uniform crossover* (UX) but, contrary to UX, it also alters the allele values of genes.
- MAM: It represents a self-adaptive mutation operator whose basic idea lies in both "sniffing" for changes in population mean during the search and following the originated population drift by means of appropriately modifying the individual's chromosome. In some sense, the working principle of this operator bears some similarity to that incorporated in CMA-ES (i.e., the *Derandomized ES with covariance matrix adaptation*) [3], where the change in population mean represents one of the decisive factors affecting the behavior of the search.
- GMASS-CVM: It represents an adaptive mutation operator based on Gaussian mutation (GM) [1], [9] that adjusts mutation step sizes in dependence on the average viability of produced mutants. The viability is defined as the percentage of mutants that were able to stand the competition in the population and survive from the generation they were created to the very next one.

3. Basic Evolutionary Scheme

As stated above, MAD-RCGA is an instance of the SSGA. In Fig. 1, the underlying principles of the algorithm are formulated in a pseudo-code. Prior to approaching the description, let us denote $|\mathbf{X}|$ as the cardinality of a set $\mathbf{X}$. Based on the information provided in the previous parts of this paper, the detail structure of the algorithm looks as follows (see Fig. 1 and Fig. 2):

- After initialization (**initialize()**) and evaluation (**evaluate()**), a population of individuals $\mathbf{P}$ is formed. The matrix $\mathbf{D}_{\mathrm{MAM}}$ and the vector $\mathbf{d}_{\mathrm{GMASS\text{-}CVM}}$ are filled up with zeros and the population mean is computed (**mean()**) and stored in the vector $\mathbf{c}_{\mathrm{new}}$. Thereafter, the following steps are iteratively applied until a termination criterion is met (for clarity, consult the respective sections):
  - At first, the matrix $\mathbf{D}_{\mathrm{MAM}}$, keeping records of the population drift $\boldsymbol{\Delta}$ gathered during the last max_{MAM} generational cycles, is updated ($\mathbf{mtx}_{\mathbf{MAM}}\mathbf{()}$) (see Sub-section 6.1).
  - The vector $\mathbf{d}_{\mathrm{GMASS\text{-}CVM}}$, keeping records of the viability of mutants collected over the course of

the last $max_{\text{GMASS-CVM}}$ generational cycles, is updated (**vec$_{\text{GMASS-CVM}}$()**). The viability is defined as the percentage of mutants produced by GMASS-CVM that survived from the generation they were created to the very next one (see Sub-section 6.2). The number of survived mutants is denoted as m_{SURV} and the number of created mutants is denoted as m_{MUT}.

- Afterwards, a χ-sized set of individuals is drawn by a parent selection mechanism (PSM) from the population **P**. This set is completed by adding yet-another individual selected uniformly randomly (**sel()**) from the population **P**. All of those individuals are subsequently moved to a mating pool. In the mating pool, the individuals undergo recombination using UWX (**rec$_{\text{UWX}}$()**) in a way that is described in more details in Section 5. Applying UWX to one reproduction event produces one offspring solution.
- After completing a λ-sized set of offspring solutions, the statistical information extracted from all the vector data stored in the matrix $\mathbf{D}_{\text{MAM}}$, in terms of expectation ζ and variance σ^2, is evaluated (**cmp_stat()**) (see Sub-section 6.1).
- Thereafter, the offspring solutions are mutated using MAM (**mut$_{\text{MAM}}$()**) and GMASS-CVM (**mut$_{\text{GMASS-CVM}}$()**), respectively (see Section 6). Whereas the MAM procedure is applied to every offspring, GMASS-CVM is applied with probability p_{m}. As can be seen from Fig. 1, mutating by means of MAM is applied along with the computation of the related statistics after having finished max_{MAM} generational cycles because till then, not all the data necessary for filling up the matrix $\mathbf{D}_{\text{MAM}}$ are collected yet.
- Compute the number of mutants m_{MUT} that were created by GMASS-CVM.
- Afterwards, the offspring solutions are transferred to an intermediate population **I** where they are evaluated (**evaluate()**).
- According to a replacement policy instantiated by PES (**update$_{\text{PES}}$**()):
  - All individuals in the population **P** are selected to perish.
  - From the union of $\mathbf{P} \cup \mathbf{I}$, a $|\mathbf{P}|$-sized subset of the best-performing individuals are picked out to form a new population of parents.
- Compute the number of mutants m_{SURV} that were created by GMASS-CVM and survived into the next generation (**cmp_m$_{\text{SURV}}$()**). Compute the corresponding viability (see Sub-section 6.2).
- Copy the vector $\mathbf{c}_{\text{new}}$ to $\mathbf{c}_{\text{old}}$.
- Update the population mean (**mean()**) and store it in the vector $\mathbf{c}_{\text{new}}$.
- Compute the population drift Δ defined as a change in population mean between two successive generations i.e., $\mathbf{c}_{\text{new}} - \mathbf{c}_{\text{old}}$.
- At the end, the generational counter t is raised by one.

As shown in Fig. 1, MAD-RCGA accepts a lot of input arguments that can be regarded as user-defined constants (see Section 5 and Section 6 for their clarifications). Those are:

- The population size N_{pop}.
- The number of offspring solutions λ produced per generation.
- The number of parents χ taking part in *parameter-wise recombination* (PWX) (see Sub-section 5.1).
- The size max_{MAM} of the matrix $\mathbf{D}_{\text{MAM}}$ keeping records of the values of Δ collected over the last max_{MAM} generations (see Sub-section 6.1).
- The size $max_{\text{GMASS-CVM}}$ of the vector $\mathbf{d}_{\text{GMASS-CVM}}$ cumulating the values of viability belonging to mutants created by GMASS-CVM (see Sub-section 6.2).
- The rate p_{m} of applying GMASS-CVM to offspring solutions.
- The coefficient β and the learning rate τ that are both associated with the configuration of the MAM operator (see Sub-section 6.1).
- The upper bound s of the interval $[0, s]$ within which the global mutation step size associated with each individual is initialized uniformly randomly at the beginning of a run (see Section 4 and Sub-section 6.1).
- The vector $\mathbf{m}_{\text{step}}$, belonging to the configuration of GMASS-CVM, that is filled up with a set of initial values of mutation step sizes associated with the respective coordinate axes (see Sub-section 6.2).
- The vector $\mathbf{f}_{\text{step}}$, belonging to the configuration of GMASS-CVM that holds a set of values of reduction factors associated with the respective mutation step sizes of $\mathbf{m}_{\text{step}}$ (see Sub-section 6.2).

4. Genotype-Phenotype Mapping

As known, configuration parameters of each algorithm may significantly influence its behavioral characteristics. These parameters can be either exogenous (i.e., derived or imposed externally) or endogenous (i.e., derived or imposed internally). Except for the global mutation step size (i.e., mutation strength), belonging to the configuration of MAM, all other parameters in MAD-RCGA are exogenous or changed exogenously.

The global mutation step size is encoded on the individual's chromosome along with object variables and changed self-adaptively (i.e., endogenously) without any intervention from outside over the course of a run.

The individual's chromosome undergoes mutation, but contrary to object variables, the global mutation step size is not subjected to recombination (see Section 5 and Subsection 6.1).

In the biological terminology, the individual in MAD-RCGA is assumed to be a single-chromosome haploid organism. Its chromosome (see Fig. 3) is a vector of floating-point numbers comprising the global mutation step size s_g and object variables $x_1 \div x_n$ (n is the problem dimensionality). The items of this vector are referred to as genes and all permissible values each gene can take on are referred to as alleles. The organism's phenotype is represented by object variables.

```
MAD_RCGA(N_pop, λ, χ, max_MAM, max_GMASS-CVM, p_m, β, τ, s, m_reduction, m_step):
{
    t = 0; /* generational counter */
    Δ = 0; /* population drift */
    count_MAM = 1;
    count_GMASS-CVM = 1;
    trigger = 0;
    count_trigger = 0;
    ν_viability = 0;

    /* D_MAM and d_GMASS-CVM are initialized with zeros */
    [P(t), D_MAM, d_GMASS-CVM] = initialize(s, N_pop, max_MAM, max_GMASS-CVM);
    evaluate(P(t));
    c_new = mean(P(t));
    while not(termination criterion)
    {
      if (t > 0)
      {
         [D_MAM, count_MAM] = mtx_MAM(D_MAM, count_MAM, max_MAM, Δ);
         [d_GMASS-CVM, count_GMASS-CVM] =
          vec_GMASS-CVM(d_GMASS-CVM, count_GMASS-CVM, max_GMASS-CVM, m_MUT,...
                        ν_viability);
      }

      /* select (χ + 1) parents to mate; χ by a PSM and one at random */
      I = rec_UWX(sel(P(t)), χ + 1);

      if (t ≥ max_MAM)
      {
         [ζ, σ] = cmp_stat(D_MAM);
         I = mut_MAM(I, λ, ζ, σ, β, τ);
      }

      [I, m_MUT, m_step, count_trigger, trigger] =
       mut_GMASS-CVM(I, λ, d_GMASS-CVM, max_GMASS-CVM, count_trigger, trigger,...
                     f_reduction, m_step, p_m);

      evaluate(I);
      P(t + 1) = update_PES(I, P(t)); /*replacement - PES*/
      m_SURV = cmp_m_SURV(P(t + 1));
      if (m_MUT > 0)
      {
         ν_viability = 100*m_SURV/m_MUT;
      }
      c_old = c_new;
      c_new = mean(P(t + 1));
      Δ = c_new - c_old;

      t = t + 1;
    }
}
```

Fig. 1. Pseudo-code for the MAD-RCGA scheme.

```
mtx_MAM(D_MAM, count_MAM, max_MAM, Δ):
{
      if (count_MAM < max_MAM)
      {
         D_MAM(1, count_MAM) = Δ;
         count_MAM = count_MAM + 1;
      } else
      {
         D_MAM(1, max_MAM) = Δ;
         count_MAM = 1;
      }

      return (D_MAM, count_MAM);
}

vec_GMASS-CVM(d_GMASS-CVM, count_GMASS-CVM, max_GMASS-CVM, m_MUT, ν_viability):
{
      if (count_GMASS-CVM < max_GMASS-CVM)
      {
         if (m_MUT > 0)
         {
            d_GMASS-CVM(1, count_GMASS-CVM) = ν_viability;
            count_GMASS-CVM = count_GMASS-CVM + 1;
         }
      } else
      {
         if (m_MUT > 0)
         {
            d_GMASS-CVM(1, max_GMASS-CVM) = ν_viability;
            count_GMASS-CVM = 1;
         }
      }

      return (d_GMASS-CVM, count_GMASS-CVM);
}
```

Fig. 2. Pseudo-code for the functions taking care of filling the vector $\mathbf{d}_{\text{GMASS-CVM}}$ and the matrix $\mathbf{D}_{\text{MAM}}$.

Obviously, this type of encoding between genotype and phenotype does not represent the one-gene-one-parameter correspondence because not all genes are directly coupled to object variables i.e., to phenotypic traits.

x_1	x_2	…	x_n	s_g

Fig. 3. Details of the individual's genotype.

As already stated in Section 3, the value of the global mutation step size associated with each individual is initialized uniformly randomly within an interval of [0, *s*] at the beginning of a run. The symbol *s* acts as a user-defined constant called the *initial upper step size bound*.

The intention of directly encoding the global mutation step size on the individual's chromosome is to give up "futilely" searching for some "optimal" problem-matching settings and let the global step size self-adaptively evolve only in dependence on the demands of a particular search that may be changing over the course of a run.

5. Uniform-wise Crossover

As stated in Section 2, the purpose of UWX in the algorithm is to provide a high scope of exploration while still taking advantage of the strong selection pressure enforced by the application of PES. This high exploration scope represents one of the main factors intentionally incorporated into MAD-RCGA to offset the "exploitative" force produced by PES. UWX helps the algorithm together with PES in both adjusting and maintaining a "reasonable" balance between exploration and exploitation and thus preventing the population from collapsing prematurely.

The design of this crossover was inspired by real-valued *uniform crossover* (UX) [4] but unlike UX, the values of genes are modified during the process of recombination. In some sense, UWX can be regarded as an extension of real-valued UX to multi-parent recombination. Its working principle can be decomposed into these two interrelated functional parts:

- We will refer to this part as *parameter-wise crossover* (PWX). PWX represents a multi-parent vector-wise recombination operator that is responsible for the alteration of the values of all genes. It is supposed to be applied as a first. The explanation of this operator is given in Sub-section 5.1.
- Uniform crossover (i.e., its real-valued version) [4]. As known, real-valued UX represents a two-parent variable-wise recombination operator mimicking the

working principle of its binary variant. UX as part and parcel of UWX is responsible for introducing a high level of exploration into the overall recombination process. It is supposed to be applied as a second.

In other words, UWX can be thought of as a mapping composed of PWX and UX and hence, besides the symbol UWX, it will also be denoted as UX•PWX. The other denotations used throughout this section follow those given and established in Section 3.

5.1 Parameter-wise Crossover

A version of this operator was originally introduced in [8], where it was designated as *parameter-sensitive crossover* (PSX) and used as a sole recombination operator (i.e., not used in a "compound" mapping as it is in case of PWX). Since then, the original theory of PSX has been reworked as follows:

- A χ-sized set of individuals is drawn by a PSM using a sampling algorithm without replacement. From the set, one parent vector is selected uniformly randomly to act as a *pivot*. Let us denote it as $\mathbf{p}_\pi$. In this sense, the pivot serves as a point of symmetry around which potential offspring vectors are generated uniformly randomly.
- Then, the PWX operator is applied to the χ selected individuals in accordance with the relation (1). As already mentioned in Section 4, only object variables are expected to undergo the recombination process i.e., the gene encoding the global mutation step size (denoted as s_g in Fig. 3) is kept intact by this process.
- As a result of applying PWX, one "virtual" offspring vector $\mathbf{d}_{\text{virtual}}$ is created. In this context, the word "virtual" signifies that the newly created individual is not considered as an end product of applying UWX to those χ individuals but only as some intermediate product that still has to be subject to the application of UX.

The breeding process itself can mathematically be described as follows:

$$\mathbf{d}_{virtual} = \mathbf{p}_\pi + \sum_{\substack{i=1 \\ i\neq\pi}}^{\chi} t_i \cdot (\mathbf{p}_i - \mathbf{p}_\pi). \qquad (1)$$

In (1), the symbol t_i ($i \in \{1, \ldots, \chi\} \wedge i \neq \pi$) stands for a random number belonging to an interval of [-1; 1] obtained by means of a uniform random number generator.

From a geometrical point of view, virtual offspring are distributed with equal probability around the pivot within a predefined region. In this sense, PWX can be conceived as a parent-centric recombination operator. Moreover, because of its vector-wise constitution, this operator is invariant against rotations of the search space or, in other words, it is independent of the choice of a coordinate system. The region in which virtual offspring are produced is restricted to the subspace spanned by the vectors ($\mathbf{p}_i$ - $\mathbf{p}_\pi$) for $i \in \{1, \ldots, \chi\} \wedge i \neq \pi$. It means that the higher is the number of parents to mate, the larger is the subspace in which virtual offspring are created. Involving more parents in reproduction has two inversely-related consequences:

- A decrease in computational efficiency (i.e., number of evaluations needed to reach the optimum) because of an enlarged space to investigate.
- An increase in computational effectiveness (i.e., rate of success in finding the true global optimum) because of a larger exploration scope.

In other words, the larger is the space to investigate, the more exploration work has to be done by the algorithm. Therefore, the number of parents participating in recombination has to be determined with caution in order to make a compromise between the levels of exploration and exploitation.

The role of this operator in UWX is shifted to doing exploitation work rather than exploration work. Therefore, its sole usage in MAD-RCGA is not recommended because of a higher chance of breaking the balance between exploration and exploitation in favor of exploitation too early, which could inevitably lead to premature convergence.

5.2 Compound Crossover (UWX)

As mention above, the UWX operator can be thought of as a compound mapping (i.e., UWX = UX•PWX) that transforms a set of parental chromosomes to an offspring one. Prior to applying UWX to a set of parents, the following steps have to be taken:

- At first, the PWX operator is applied to a set of χ parents selected by a PSM as described in Sub-section 5.1. As a result, a virtual offspring $\mathbf{d}_{\text{virtual}}$ is made.
- Thereafter, one additional parent $\mathbf{p}_{\text{additional}}$ picked out uniformly randomly from the population is mated with $\mathbf{d}_{\text{virtual}}$ in order to form a "final" offspring. This mating procedure is accomplished by the UX operator applied to the genes encoding object variables. The value of the gene associated with the global mutation step size is simply copied from $\mathbf{p}_{\text{additional}}$ to that final offspring.

Because of the strict variable-wise nature of the UX operator, its application to an end product of PWX yields a loss of invariance against rotations of the search space. On the other hand, this negative effect is partially compensated by introducing a higher level of exploration into UWX.

6. Mutation in MAD-RCGA

In this section, two types of mutation operators used in MAD-RCGA will be described in more detail. As stated

in Section 2, those are MAM and GMASS-CVM. The purpose of these operators in the algorithm is twofold:

- To provide a "traditional" way of disturbing a gene in the individual's chromosome in order to promote the population diversity by injecting new alleles. The GMASS-CVM operator is in charge of this functionality in MAD-RCGA.
- To follow the population drift that develops as a consequence of applying evolution operators. In this sense, the heuristics hidden behind MAM is laid down on the presumption that the population has a tendency of drifting towards the optimum (or some promising regions). Based on this, the drift is supposed to be the most probable direction in which the individual's chromosome should be altered in order to yield a maximal profit to the fitness. The MAM operator is responsible for taking care of this type of functionality in MAD-RCGA. In some sense, following the population drift can also be regarded as a way of incorporating linkage among genes. In the light of this fact, promoting diversity is conceived as a secondary effect of applying MAM.

Throughout the following subsections, the notation $w \sim N(0, \sigma)$ denotes a realization of a zero-mean normally distributed random variable with variance σ^2 and the symbol n stands for the problem dimensionality. The other used denotations follow those given and established in Sec. 3.

6.1 Mean-Adaptive Mutation

MAM can be characterized as a self-adaptive linkage-preserving mutation operator harnessing a movement in population mean developing during the search process as a consequence of application of evolution operators. The heuristic behind MAM is built upon the presumption that this movement represents a drift towards promising regions. In this regard, it is considered to be the most likely direction of modifying the individual's chromosome being expected to yield a maximal profit to the fitness.

To that end, the changes in population mean that arose during the last max_{MAM} generations are recorded and kept in the dedicated matrix $\mathbf{D}_{\mathrm{MAM}}$ (see Fig. 2). This matrix represents a container of vectors wherein the oldest data is replaced with the latest at each generation. The data kept in $\mathbf{D}_{\mathrm{MAM}}$ is statistically processed, in terms of expectation $\boldsymbol{\zeta}$ (2) and standard deviation $\boldsymbol{\sigma}$ (3), by the **cmp_stat()** function (see Section 3 and Fig. 1) as follows[1]:

$$\boldsymbol{\zeta} = \frac{\sum_{j=1}^{max_{\mathbf{MAM}}} \mathbf{D}_{\mathbf{MAM}}(j)}{max_{\mathbf{MAM}}}, \tag{2}$$

$$\boldsymbol{\sigma} = \sqrt{\frac{\sum_{j=1}^{max_{\mathbf{MAM}}} \left(\mathbf{D}_{\mathbf{MAM}}(j) - \boldsymbol{\zeta}\right)^2}{\left(max_{\mathbf{MAM}} - 1\right)}}. \tag{3}$$

In the above equations, the symbol $\mathbf{D}_{\mathrm{MAM}}(j)$ stands for the j-th vector component (i.e., a population drift vector) of $\mathbf{D}_{\mathrm{MAM}}$. The expectation $\boldsymbol{\zeta}$ indicates an average tendency in population mean developed over the last max_{MAM} generations (i.e., an average value of the population drift).

Having evaluated the expectation $\boldsymbol{\zeta}$ and standard deviation $\boldsymbol{\sigma}$, the procedure performed by MAM can be written as follows (in order of execution):

- At first, the vector $\mathbf{v}_{\mathrm{path}}(j)$, determining the direction of chromosomal modifications, is computed according to the relation (4). Herein the coefficient β serves as a reduction factor mitigating the level of disturbance introduced by the standard deviation.

$$\begin{aligned} &\mathbf{v}_{\mathbf{path}}(j) = \boldsymbol{\zeta} + \beta \cdot \mathbf{w}(j), \\ &w_k(j) \sim N(0, \sigma_k), \\ &j \in \{1, \ldots, N_{pop}\}, k \in \{1, \ldots, n\}. \end{aligned} \tag{4}$$

- Thereafter, the global step size $s_j(g)$ at generation g encoded on the chromosome of the j-th individual is adapted. The adaptation is laid down on the concept of *mutative strategy parameter control* (MSC) used in ESs ([1], [3], [9]). It means that the global step size $s_j(g)$ as part and parcel of the individual's chromosome undergoes mutation according to (5) prior to its exploiting for mutating the object variables in a vector-wise manner according to (6). Mutation on the level of $s_j(g)$ is performed by using a lognormal probability distribution just as in ESs. The corresponding mutation strength[2] τ is kept constant over the search and is regarded as a user-defined constant.

$$\begin{aligned} &s_j(g+1) = s_j(g) \cdot \exp(\eta_j), \\ &\eta_j \sim N(0, \tau), \\ &j \in \{1, \ldots, N_{pop}\}. \end{aligned} \tag{5}$$

- At the end, $\mathbf{v}_{\mathrm{path}}(j)$ is used for mutating the chromosome of the j-th individual, given by the vector $\mathbf{d}(j)$, according to (6). The strength of mutation on the object variable level is governed by the global mutation step size $s_j(g+1)$. The notation $u_j \sim U(0, s_j(g+1))$ used in the relation (6) denotes a realization of a uniformly distributed random variable within a continuous interval of $[0; s_j(g+1)]$.

$$\begin{aligned} &\mathbf{d}(j) = \mathbf{d}(j) + u_j \cdot \mathbf{v}_{\mathbf{path}}(j), \\ &u_j \sim U(0, s_j(g+1)), \\ &j \in \{1, \ldots, N_{pop}\}. \end{aligned} \tag{6}$$

[1] Mathematical operations with vectors such as square root and power of two are performed in a component-wise manner.

[2] Also called *learning rate*.

```
mut_MAM(I, λ , ζ, σ , β, τ):
{
      /* for each offspring from the intermediate population I */
      for j = 1 to λ
      {
         /* retrieve the chromosome of the j-th individual from the
            intermediate population I; the chromosomal structure is of
            the form d = (x1, x2, ... , xn, sg) – see also Fig. 3 */
         d(1:n+1) = I(j);
         /* retrieve the global step size from the chromosome */
         sg = d(n+1);
         /* generate a zero-mean normally distributed random number */
         η = randn(0, τ);
         /* mutate the global step size according to (5) */
         sg = sg*exp(η);
         /* generate an n-dimensional vector of zero-mean normally
            distributed random numbers */
         w(1:n) = randn(0, σ(1:n));
         /* generate the corresponding path vector according to (4) */
         vpath(1:n) = ζ(1:n) + β*w(1:n);
         /* generate a uniformly distributed random number */
         u = rand(0, sg);
         /* mutate the chromosome d according to (6) */
         d(1:n) = d(1:n) + u*vpath(1:n);
         /* add the modified step size back to the individual's chromosome d */
         d(n+1) = sg;
         /* add the modified chromosome back to the intermediate
            population I */
         I(j) = d;
      }

      return (I); /* return the modified offspring population */
}
```

Fig. 4. Pseudo-code for the function performing the MAM procedure.

Contrary to ESs that take advantage of a normal probability distribution, this algorithm makes use of a uniform probability distribution for mutation on the object variable level. The reasons are as follows:

- To conform to the fundamental idea behind MAM as regards preserving the sense of chromosomal modifications identical to the direction of $\mathbf{v}_{path}(j)$.
- Limiting the strength of mutation on the object variable level is observed to have favorable impacts on performance. Any "extreme" out-of-interval modifications that could otherwise occur in case of a normal probability distribution might grind the evolution to a halt (or at least slow it down).

One of the possible implementations of the MAM procedure is outlined in a pseudo-code in Fig. 4.

To recap, the concept of self-adaptation used in MAM is built upon the following two ideas:

- Harnessing the changes in population mean, developing as a consequence of application of evolution operators, for driving the search towards promising regions.
- Evolving the global step size s_g (see Fig. 3) in accordance with the concept of MSC in order for the strength of mutation on the global step size level to be adjusted endogenously to the current topology of the fitness function.

Because every idea incorporated into the working principle of an algorithm represents a kind of domain knowledge restricting the width of its applicability (the No free lunch theorem [10] confirms this fact), it is worth highlighting and summarizing some of the obvious merits and demerits of the MAM approach:

- Mutating the chromosome in a vector-wise manner with respect to the changes in population mean aids in both preserving linkage among variables and alleviating the dependency of the algorithm on the choice of a coordinate system. In this way, both the adverse effects of UWX on linkage preservation and the loss of invariance of UWX against rotations of the search space are partly compensated.
- There is safety in numbers i.e., the presence of multiple individuals in the population makes the estimates of the expected population drift more accurate. This results in growing computational effectiveness and efficiency.
- In order for MAM to yield some improvements to the algorithm, it is assumed that individuals in the popu-

lation will start behaving like a "mass" of indistinguishable elements i.e., in a "similar" fashion to manifest a systematic drift. However, this type of behavior is possible only if there is some detectable regularity in the fitness landscape.

- The previous point signifies that the area of applicability of MAM could preferentially lie in the following types of fitness landscapes:
  - Unimodal functions – in this case, the "regularity" instigating the population drift is ensured by the presence of the only attractor.
  - Multimodal functions with such a type of regularity that is able to originate a systematic and measurable drift in population mean. Typically, those are multimodal functions having either their extremes regularly scattered over the search space or some global attractor with a large basin.

  In conclusion, it is necessary to say that by no means the absence of any regularity in the fitness landscape implies the inability of the algorithm to find some acceptable solutions. It only means that the usage of MAM has probably no or negligible effect on improving the overall performance of the algorithm. The extent to which the absence of a systematic drift in the population affects the performance of the algorithm depends on the character of a problem itself.

In the paper [2], Beyer and Deb argue that the population mean should be preserved after having applied variation operators (i.e., mutation or/and recombination). In our opinion, this postulate is too restrictive. We think that a variation operator is allowed to change the population mean but only under condition that such a change is made endogenously without any a priori built-in bias as a reaction to the actual state of evolution in the algorithm. MAM is an example of such a behaving operator i.e., an operator whose bias is not a result of some a priori built-in piece of information but the evolution process itself.

6.2 Gaussian Mutation with the Adaptive Step Size Control Based upon the Viability of Produced Mutants

In contrast to MAM, this mutation operator offers a more traditional way of altering the individual's chromosome being based on GM [1], [9]. The adaptation process consists in adjusting the values of mutation step sizes in dependence on the viability of produced mutants, where the word *mutant* denotes a chromosomal alteration that is incurred by application of the GMASS-CVM operator.

Viability is defined as the percentage of mutants that survived from the generation they were produced into the very next one. It can be quantified as follows:

$$\nu_{viability}(g) = 100 \cdot \frac{m_{SURV}(g+1)}{m_{MUT}(g)}. \tag{7}$$

In Equation (7), $m_{\mathrm{MUT}}(g)$ denotes the number of mutants created at generation g and $m_{\mathrm{SURV}}(g+1)$ denotes the number of the mutants that survived into the very next generation, i.e., $(g + 1)$. The symbol $\nu_{\mathrm{viability}}(g)$ stands for the viability (or successfulness) of mutants that were emitted into the population at generation g.

In this context, we also define so-called *average viability* as the expectation of all the values of viability collected over the last $max_{\mathrm{GMASS\text{-}CVM}}$ generations. The value of $max_{\mathrm{GMASS\text{-}CVM}}$ significantly affects the duration of the exploration phase (i.e., the larger is the value, the longer is the exploration period). Throughout the search process, the values of viability are continually recorded and kept in the dedicated vector $\mathbf{d}_{\mathrm{GMASS\text{-}CVM}}$ (see Fig. 2). The vector represents a container wherein the oldest data is replaced with the latest at each generation. The average viability at generation g is computed as follows:

$$\underline{\nu}_{viability}(g) = \frac{\sum_{j=1}^{max_{\mathbf{GMASS-CVM}}} d_{GMASS-CVM}(j)}{max_{\mathbf{GMASS-CVM}}}. \tag{8}$$

The evaluation is carried out in the function **expectation()** depicted in Fig. 5. In (8), $d_{\mathrm{GMASS\text{-}CVM}}(j)$ denotes the j-th scalar component of $\mathbf{d}_{\mathrm{GMASS\text{-}CVM}}$ (i.e., a value of viability). The value of $\underline{\nu}_{\mathrm{viability}}(g)$, standing for the average viability at generation g, is computed from all the values of $\nu_{\mathrm{viability}}(k)$ where $k \in \{g–max_{\mathrm{GMASS\text{-}CVM}}+1, g–max_{\mathrm{GMASS\text{-}CVM}}+2,\ldots, g\}$.

The steps that are to be taken by this operator in order to fulfill its task are performed within **mut$_{\mathbf{GMASS\text{-}CVM}}$()** depicted in Fig. 1. The working principle itself can be described as follows (in order of execution – see Fig. 5 for details):

- If $\mathbf{d}_{\mathrm{GMASS\text{-}CVM}}$ is filled up with the values of viability associated with the current values of mutation step sizes, set the *trigger* flag.
- If the trigger flag is set, compute the value of $\underline{\nu}_{\mathrm{viability}}(g)$[3] according to the relation (8) and test it for zero . If zero, reduce the values of mutation step

[3] In GMASS-CVM, the value of $\underline{\nu}_{\mathrm{viability}}(g)$ plays the role of an average measure of successfulness of mutation as to the current values of mutation step sizes (i.e., one step size per coordinate axis). The values of $\nu_{\mathrm{viability}}(g)$ associated with the current mutation step sizes are at first culled over $max_{\mathrm{GMASS\text{-}CVM}}$ generations until the vector $\mathbf{d}_{\mathrm{GMASS\text{-}CVM}}$ is full. After completing $\mathbf{d}_{\mathrm{GMASS\text{-}CVM}}$ with the new data (i.e., after passing these $max_{\mathrm{GMASS\text{-}CVM}}$ generations), the *trigger* flag from Fig. 5 is set. Having set this flag signifies that the process of collecting $\nu_{\mathrm{viability}}(g)$ will be extended by calculating the value of $\underline{\nu}_{\mathrm{viability}}(g)$ at each subsequent generation in order to test it for zero.

sizes[4] that are kept in the dedicated vector $\mathbf{m}_{\text{step}}$[5]. The reduction of mutation step size values is accomplished by dividing $\mathbf{m}_{\text{step}}$ by the vector $\mathbf{f}_{\text{reduction}}$ in a component-wise manner. The items of $\mathbf{f}_{\text{reduction}}$, called *reduction factors*, represent user-defined parameters that are kept constant over the search process.

- In case the adjustment of mutation step sizes has been made, the trigger flag from Fig. 5 is unset. It means that prior to launching again the process of computing and testing $\underline{\nu}_{\text{viability}}(g)$ for zero, the vector $\mathbf{d}_{\text{GMASS-CVM}}$ has to be completely updated by values of $\nu_{\text{viability}}(g)$ for these new adjusted step sizes.
- The mutation procedure itself is accomplished by selecting an individual to be mutated with probability p_{m} and altering the value of its uniformly randomly selected gene. The strength of mutation is given by the value of the step size associated with that gene. Let us suppose that d_{i} is the uniformly randomly selected gene. Then the alteration of the gene can be written as follows:

$$d_i' = d_i + w, \qquad (9)$$
$$w \sim N\left(0, m_{step}(i)\right).$$

  In (9), the symbol $m_{\text{step}}(i)$ stands for the i-th scalar component (i.e., the i-th mutation step size) of $\mathbf{m}_{\text{step}}$.
- The number of mutants created at each generation is counted and the result is kept recorded in the variable m_{MUT} whose value is necessary to know for evaluating the corresponding viability as to relation (7).

The exploitation of a normal probability distribution for disturbing the individual's chromosome is a very stimulating factor for the resultant quality of the search, as regards exploration, because of assigning each point in the search space a non-zero probability of being reachable by means of mutation perturbations only. This lends the operator the ability of producing alterations from beyond the horizon limited by the actual values of mutation step sizes. At the beginning of the search, when the values of mutation step sizes are maximal, this operator has bias in favor of doing more exploration work. As the search advances, mutation step size values become smaller and smaller, and the operator increasingly favors exploitation up to local tuning.

At the end of each generation, the mutants created by GMASS-CVM compete with other individuals for places in the population. In order for them to survive, they have to represent viable modifications as regards fitness. In other words, it means that a mere alteration of the value of one randomly-selected gene has to cause such an increase in fitness that will be sufficient for the relevant individual to withstand the competition and survive into the next generation. This fact predetermines the application area[6] of the operator especially to those fitness functions that are "sensitive" to changes in one of their parameters i.e., to those having no or low epistasis.

7. Conclusions

In the first part of the paper, we have presented theoretical foundations of a novel, promising instance of the RCGA, called the Mean-adaptive RCGA (MAD-RCGA), which was designed to aid in addressing hard-to-solve technical optimization problems. This instance incorporates three novel evolution operators lending the algorithm a large exploration scope and the ability to self-control its own mutability as the search advances. Moreover, it also takes advantage of population-elitist selection (PES), adopted from evolution strategies, which is responsible for producing a sufficiently high level of selection pressure that warrants a reasonable convergence velocity.

Acknowledgements

The research described in the paper was financially supported by the Czech Grant Agency under grant No. 102/07/0668, by the COST project IC0603 ASSIST, and by the research program MSM 0021630513.

References

[1] BÄCK, T., SCHWEFEL, H. P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation Journal*. The MIT Press, 1993, vol. 1, no. 1, p. 1–23.

[2] BEYER, H. G., DEB, K. On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation,* 2001, vol. 5, no. 3, p. 250–270.

[3] HANSEN, N., OSTERMEIER, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*

[4] The current values of mutation step sizes are reduced in anticipation of increasing the viability of mutants produced by them in the next generations. In this way, mutation step size values are decreased adaptively and irreversibly as the search process runs. It is necessary to note that averaging the values of viability for given values of mutation step sizes over a predefined number of generations is done with the intention of reducing the probability of some occasional fluctuations that could otherwise lead to their rapidly and inadequately lowering.

[5] The vector $\mathbf{m}_{\text{step}}$ is set up with respect to the expected scope of the exploration phase at the beginning of a run. The values of $\mathbf{m}_{\text{step}}$ are expected to evolve as the search process advances in dependence on the viability of produced mutants.

[6] I.e., the application area where the operator yields a maximal profit to the algorithm.

Journal. The MIT Press, 2001, vol. 9, no. 2, pp. 159 - 195.

[4] HERRERA, F., LOZANO, M., SÁNCHEZ, A. M. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems*. Wiley Periodicals, Inc., 2003, vol. 18, no. 3, p. 309–338.

[5] LOZANO, M., HERRERA, F., CANO, J. R. Replacement strategies to preserve useful diversity in steady-state genetic algorithms. *Information Sciences*, in press (2006).

[6] LOZANO, M., HERRERA, F., CANO, J. R. Replacement strategies to maintain useful diversity in steady-state genetic algorithms. In *Proceedings of the 8th Online World Conference on Soft Computing in Industrial Applications*, September 2003.

[7] OTEVŘEL, V., OLIVA, L. Comparison of real-coding genetic algorithm with particle swarm optimization on the bandgap bandwidth maximization problem. In *Proceedings of the 17th International Conference Radioelektronika 2007*. Brno: Brno University of Technology, 2007, p. 655–658.

[8] OTEVŘEL, V., RAIDA, Z. Wide band global optimization of microstrip line using novel polytope and real genetic algorithms. In *Proceedings of the Int. Conference on Electromagnetics in Advanced Applications ICEAA'03*. Polytecnico di Torino, 2003, p. 131–134.

[9] PÉRIAUX, J., WINTER, G. (editors) *Genetic Algorithms in Engineering and Computer Science*. John Wiley & Sons Inc., 1995.

[10] SURRY, P. D. *A Prescriptive Formalism for Constructing Domain-Specific Evolutionary Algorithms*. Dissertation thesis. University of Edinburg, 1998.

[11] VINCENT, J. *Numerical Optimization using Genetic Algorithms*. Technical Report (available from http://dec.bournemouth.ac.uk/staff/jvincent/ec/numerical_optimisation_using_genetic_algorithms.pdf). University of Bournemouth, UK, 2003.

```
mut_GMASS-CVM(I, λ, d_GMASS-CVM, max_GMASS-CVM, count_trigger, trigger, f_reduction, m_step, p_m):
{
    if (count_trigger == max_GMASS-CVM)
    {
       /* set the flag if the whole vector d_GMASS-CVM is filled up with the values of
          viability for the currently used mutation step sizes */
       trigger = 1;
    }

     /* if d_GMASS-CVM is filled up, test the average viability associated with the
       currently used mutation step size values for zero */
    if (trigger == 1)
    {
       ν_avg_viability = expectation(d_GMASS-CVM); /* compute the average viability */
       if (ν_avg_viability == 0) /* if it is zero, trigger the reduction event */
       {
           m_step = m_step/f_reduction; /* reduce the respective mutation step sizes */
           trigger = 0; /* unset the flag */
           count_trigger = 0;  /* reset the counter */
       }
    }

    m_MUT = 0;
    for j = 1 to λ /* for each offspring in the intermediate population */
    {
        p = rand(0, 1); /* generate uniformly randomly a value from an interval of
                           [0, 1] */
        if (p < p_m) /* if the mutation event is instigated */
        {
          /* retrieve the chromosome of the j-th individual from the intermediate
             population I; the chromosomal structure is depicted in Fig. 3 */
          d = I(j);
          /* select uniformly randomly a gene to be mutated from the set {1, ... , n} */
           k = rand({1, ... ,n});
          /* generate a zero-mean normally distributed random number and alter the
             k-th gene of the j-th offspring according to (9) */
           w = randn(0, m_step(k));
           d(k) = d(k) + w;
           I(j) = d; /* add the chromosome back to the intermediate population */
           m_MUT = m_MUT + 1; /* increase the number of created mutants */
        }
    }

    if (m_MUT > 0)
    {
       count_trigger = count_trigger + 1; /* increase the counter if at least one
                                             mutant is created */
    }

    return (I, m_MUT, m_step, count_trigger, trigger);
}
```

Fig. 5. Pseudo-code for the function implementing the GMASS-CVM functionality.