Contents

Introduction

This book is a continuation of the *Genetic Algorithms Reference* series, the first volume of which was devoted to crossover operators. The current, second volume presents mutation operators destined to solve numerical optimization problems.

The layout of this book follows the layout of the former volume, however, the simple division into operators dedicated to solving problems coded with binary and real numbers was insufficient this time. Research work on the mutation operator does not lead only to development of its new, frequently dedicated form. One of the main questions asked in connection with this operator is about the probability of its application and about the scale of perturbation introduced by this operator. Therefore, three out of seven chapters of this book are devoted to this issue. Moreover, following the readers' opinions as well as suggestions of the reviewer of the first volume, I decided to add a chapter which is an introduction to the genetic algorithms theory. It is directed at everyone who starts *their adventure* involving genetic algorithms with this book series.

Eventually the layout of this book is as follows. The first chapter provides a general introduction to the genetic algorithms theory. The following three chapters present suggestions, formulas and methods applied in determining values of parameters controlling the mutation process. Here I have applied the most popular division[1] into: static and deterministic approach, dynamic adaptive approach and self-adaptive approach. Two of the last chapters present mutation operators developed with problems coded with binary and real numbers in mind.

As in the first volume, the division into methods dedicated to solving problems coded with binary and real numbers is somewhat artificial and most often results from the content of the source material. However, since many of the presented suggestions, rules and methods may be applied to different classes of problems, hence each of them is marked by one or many of the following symbols determining the scope of their application:

- (B) – problems coded with binary numbers;
- (R) – problems coded with real numbers;
- (D) – problems coded with integer numbers;
- (P) – determining the value of probability of mutation;
- (MS) – determining the length of mutation step;
- (MP) – determining the number of mutation points.

With the exception of the first chapter, the applied pattern of presentation is the same as in the first volume. Depending on whether a suggestion, a rule or a full method is presented, the pattern includes all or only some of the following elements:

[1] see: R. Hinterding, Z. Michalewicz, and A. E. Eiben (1997). *Adaptation in evolutionary computation: A survey.* http://citeseer.ist.psu.edu/hinterding97adaptation.html

- Keywords – are supposed to help with searching through the book and also with mutual association of the presented in it operators.
- Motivation – showing the motivation which was the base for development of a given operator. This motivation has been formulated by the authors explicit or has been drawn up arbitrarily.
- Source text – source text pointing to the website from which that text may be downloaded – most of these sites are free of charge.
- Read also – suggested additional texts, the subject matter of which is directly connected with the discussed operator. The choice of these texts, even though it is made arbitrarily, is based mainly on the bibliography list included in the source text or points to the texts describing further development of a given operator or other operators connected with it ideologically. Links to sites where the suggested texts may be downloaded from are also provided.
- See also – other operators that in my opinion it is worth to become acquainted with in connection with a given operator.
- Algorithm – presents the discussed operator in the form of a pseudo-code, often in a couple of options. I decided to choose this form of presentation of an operator because it enables, in most cases, immediate application of that operator in practice. On the other hand I decided against usage of a specific programming language because elements appearing in the code additionally, resulting from grammar could make it difficult to understand the presented operator. A presented algorithm may often differ from its original form presented in the source text, it is often the case when the form of an operator was closely connected with the problem for which a given operator has been developed. However, the key idea of an operator is always presented.
- Comments – commentary or description of the presented operator, depending on whether in my opinion pseudo-code of an algorithm is a sufficient description or not.
- Experiment domains – problems that a given operator has been developed to solve or has been tested on, especially in consideration with standard testing functions.
- Compared to – list of operators or methods the presented operator has been compared to (in the source text).

Even though it may be disapproved of, I treat the following terms as synonyms: "recombination–crossover", "solution vector–chromosome", "gene–variable", "generation–iteration" and use them as such throughout the text. Moreover, if it is not indicated explicit to be otherwise, I use following symbols throughout the text:

t – generation/iteration counter (time)
M – maximum number of generations/iterations
P() – population of solution vectors (chromosomes)
P(*t*) – current population
P(*t*+1) – next population
Population_size – number of solution vectors (chromosomes) in *P*( )
p_c – crossover probability
$p_c^{(t)}$ – crossover probability in generation *t*
$p_c^{(t)}(X)$ – crossover probability of vector *X* in generation *t*

p_m – mutation probability[2]
$p_m^{(t)}$ – mutation probability in generation t
$p_m^{(t)}(X)$ – mutation probability of vector X in generation t
n – length of solution vector (chromosome)
$N(\,)$ – Normal/Gaussian probability distribution
$C(\,)$ – Cauchy probability distribution
$U(\,)$ – Uniform probability distribution
$\lfloor a \rfloor$ – integer part of a
Rnd – random real number , $Rnd \sim U(0,1)$, if Rnd appears in an algorithm loop, then in every cycle of that loop the value of Rnd is determined again

Binary-coded operators
$A^{(t)} = (a_1^{(t)},..., a_n^{(t)})\ \forall i\ a_i^{(t)} \in \{0,1\}$ – binary-coded solution vector (chromosome)
$f(A^{(t)})$ – fitness of binary-coded solution vector $A^{(t)}$
$\{A_1^{(t)},..., A_k^{(t)}\} \in P(t)$ – binary-coded solution vectors (chromosomes)
$A_j^{(t)} = (a_{j1}^{(t)},..., a_{jn}^{(t)}) \forall i, j\ a_{ji}^{(t)} \in \{0,1\}$
$f(A_j^{(t)})$ – fitness of binary-coded solution vector $A_j^{(t)}$

Real-coded operators
$X^{(t)} = (x_1^{(t)},..., x_n^{(t)}) \in R^n$ – real-coded solution vector (chromosome)
$\forall i\ x_i^l \le x_i \le x_i^u$ where:

x_i^l – lower boundary of i^{th} variable (gene),
x_i^u – upper boundary of i^{th} variable (gene)

$f(X^{(t)})$ – fitness of real-coded solution vector $X^{(t)}$
$\{X_1^{(t)},..., X_k^{(t)}\} \in P(t)$ – real-coded solution vectors (chromosomes)
$\forall i, j\ X_j^{(t)} = (x_{j1}^{(t)},..., x_{jn}^{(t)}) \in R^n,\ x_i^l \le x_{ji} \le x_i^u$ where:

x_i^l – lower boundary of i^{th} variable (gene)
x_i^u – upper boundary of i^{th} variable (gene)

$f(X_j^{(t)})$ – fitness of real-coded solution vector $X_j^{(t)}$

[2] The term *probability of mutation* in the GA theory almost always means probability of mutation of a single gene in a chromosome and this will be the approach in the entire book. Cases, where this term will concern the whole chromosome (and not a single gene) will be clearly emphasized in the content of this book.

Upon publishing of whole series of the planned books, I plan to prepare supplements (in e-book format) once every two years which will cover two years from the date that book was published. I plan to send them to all interested readers. Hence, if you wish to receive such a supplement in the future please contact me by e-mail.