Center of Mass Encoding: A Self-Adaptive Representation with Adjustable Redundancy for Real-Valued Parameters

Claudio Mattiussi
Ecole Polytechnique Fédérale de Lausanne (EPFL) Laboratory of Intelligent Systems, CH-1015 Lausanne, Switzerland
http://lis.epfl.ch
claudio.mattiussi@epfl.ch

Peter Dürr
Ecole Polytechnique Fédérale de Lausanne (EPFL) Laboratory of Intelligent Systems, CH-1015 Lausanne, Switzerland
http://lis.epfl.ch
peter.duerr@epfl.ch

Dario Floreano
Ecole Polytechnique Fédérale de Lausanne (EPFL) Laboratory of Intelligent Systems, CH-1015 Lausanne, Switzerland
http://lis.epfl.ch
dario.floreano@epfl.ch

ABSTRACT

In this paper we describe a new class of representations for real-valued parameters called Center of Mass Encoding (CoME). CoME is based on variable length strings, it is self-adaptive, and it permits the choice of the degree of redundancy of the genotype-to-phenotype map and the choice of the distribution of the redundancy over the space of phenotypes. We first describe CoME and then proceed to test its performance and compare it with other representations and with a state-of-the-art evolution strategy. We show that CoME performs well on a large set of test functions. Furthermore, we show how CoME adapts the granularity of its discretization on functions defined over nonuniformly scaled domains.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Performance, Experimentation

Keywords

CoME, center of mass encoding, genetic encoding, genetic algorithms, evolutionary algorithms, redundant representation, adaptive representation, real parameters

1. INTRODUCTION

In many applications of evolutionary computation there is the need to represent real-valued parameters. A common approach is to represent each parameter as a binary string of fixed length, interpret the string as a binary or a Gray-coded integer, and map the integer linearly or exponentially to the desired real interval [2, 5]. A generalization of this approach is based on the use of a β-ary alphabet with $\beta > 2$ in lieu of the binary alphabet [8]. These representations give the user the flexibility of defining the represented interval and the granularity of its quantization. However, this flexibility is a mixed blessing, since the user must decide how many bits to use for each parameter, with the risk of working with a quantization that is too coarse for the problem at hand if too few bits are used, and the complementary risk of increasing too much the size of the search space if too many bits are used. For this reason, some authors have suggested techniques that adapt the representation during the course of the evolutionary runs. For example, Schraudolph and Belew [9], Kwon *et al.* [4], and Streifel *et al.* [10] proposed algorithms that periodically redefine the mapping of fixed-length binary strings by changing the target interval of the representation according to the history of the run. Other techniques follow an iterative approach. Whitley *et al.* [12] suggest a strategy that relaunches a genetic algorithm (GA) multiple times and remaps the search space to a hypercube around the solution found by the previous run.

The adaptive representations mentioned above are all based on fixed-length strings and require the explicit definition of a policy of change of the genotype-to-phenotype map based on the fitness values of the population. Moreover, for all these representations – as in the conventional binary and Gray coded representations – the genotype-to-phenotype map is one-to-one. However, it is well known that the presence of a certain degree of redundancy in the genotype-to-phenotype map can be beneficial to the evolutionary process, provided the "good" phenotypes are not grossly underrepresented [8]. Examples of redundant representation of parameters can be found in [14, 5]. In this paper we describe a new class of representations for real-valued parameters called Center of Mass Encoding (CoME). CoME is based on variable length strings, it is self-adaptive, and it permits the choice of the degree of redundancy of the genotype-to-phenotype map and the choice of the distribution of the redundancy over the space of phenotypes. We will first define CoME and then proceed to test its performance and compare it with the state-of-the-art representations for real-valued function optimization.[1]

[1]The computer code implementing CoME can be downloaded at the web address http://lis.epfl.ch/software

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

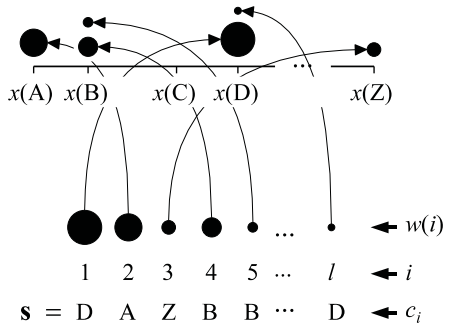

Figure 1: An example of mapping from a string s to the system of particles whose center of mass is the value of the parameter encoded by s. Note that the points $x(A), x(B), \ldots$ are not necessarily equispaced along the interval.

2. PARAMETER REPRESENTATION

Our aim is to define a function that maps a sequence of characters (i.e., a string) $\mathbf{s} = c_1 c_2 \ldots c_l$ belonging to an alphabet A to a finite real interval. Without loss of generality we can work with the interval $I = [0,1]$, with the assumption that the representation defined on I will be linearly or nonlinearly mapped to the actual interval of interest. The basic idea of CoME is to use the string $\mathbf{s}$ to define a system of particles on I and to take the center of mass of this system as the point represented by $\mathbf{s}$. Each character c_i in the string is associated with a particle whose position $x(c_i) \in I$ is determined by the value of the character and whose mass $w(i)$ is determined by the position i of the character in the string (Figure 1). The point $x(\mathbf{s}) \in I$ to which the string is mapped is thus given by

$$x(\mathbf{s}) = x(c_1 c_2 \ldots c_l) = \frac{\sum_{i=1}^{l} x(c_i)\, w(i)}{\sum_{i=1}^{l} w(i)} \qquad (1)$$

where l is the length of the string. This definition results in a large variety of mappings, depending on the choice of the alphabet and of the functions $x(c_i)$ and $w(i)$. Let us consider some examples. To simplify the notation we assume that the alphabet corresponds to the set of integers $\{0, \ldots, \beta - 1\}$, where β is the cardinality of the alphabet.

EXAMPLE 1. Totalistic encoding*: The particles are distributed evenly on the interval by setting $x(c_i) = c_i/(\beta - 1)$ and all the particles are given the same mass $w(i) = \mu$. We obtain*

$$x(\mathbf{s}) = \frac{\sum_{i=1}^{l} c_i}{(\beta - 1)\, l} \qquad (2)$$

For arbitrary alphabets the resulting point depends only on the sum of the values of the characters in the string (borrowing the term from the terminology of cellular automata we call this representation totalistic*). For a binary alphabet this corresponds to the unary encoding analyzed by Rothlauf [8], which maps a binary string to a point that depends only on the number of* 1*s in the string. An example of the distribution of the redundancy of this representation is shown in Figure 2.*

EXAMPLE 2. Non-redundant positional encoding*: As in Example 1 we set $x(c_i) = c_i/(\beta - 1)$ but now the particle masses are defined by $w(i) = \beta^{-i}$. We obtain*

$$x(\mathbf{s}) = \frac{\sum_{i=1}^{l} c_i\, \beta^{-i}}{(\beta - 1) \sum_{i=1}^{l} \beta^{-i}} \qquad (3)$$

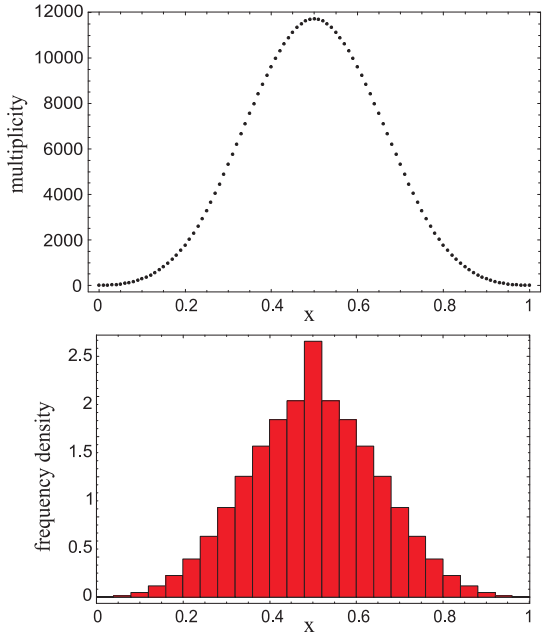

Figure 2: The totalistic encoding is characterized by a highly nonuniform distribution of the redundancy. The top plot shows the number of times a given real value is obtained (i.e., its multiplicity) with Equation (2) by mapping all the strings of length $l = 4$ over an alphabet of cardinality $\beta = 26$. The bottom histogram shows the frequency density of the values across the represented interval.

This mapping corresponds to the conventional β-ary positional representation mentioned in the introduction, which interprets the string as an integer in base β and maps it linearly to the interval I. The mapping from the set of strings to the real values is one-to-one.

EXAMPLE 3. Redundant positional encoding*: Everything remains as in Example 2, except that the base of the exponential that defines the masses can be smaller than the cardinality of the alphabet, that is, we have $w(i) = \alpha^{-i}$ with $\alpha \leq \beta$. A general genotype-to-phenotype map that includes the previous cases is*

$$x(\mathbf{s}) = \frac{\sum_{i=1}^{l} c_i\, \alpha^{-i}}{(\beta - 1) \sum_{i=1}^{l} \alpha^{-i}} \quad , \quad 1 \leq \alpha \leq \beta \qquad (4)$$

For $\alpha < \beta$ the mapping defined by Equation (4) *is redundant, as illustrated in Figure 3. For $\alpha = 1$ this mapping reduces to Equation* (2) *(totalistic encoding) and for $\alpha = \beta$ it reduces to Equation* (3) *(non-redundant positional encoding).*

The redundant positional encoding considered in Example 3 is particularly interesting because by using a value of α that is not too small with respect to the cardinality β of the alphabet, its degree of redundancy can distributed evenly across the represented domain. Note, for example, how the frequency distribution of the values becomes more uniform across the interval when the value of the base is increased from $\alpha = 1$ (Figure 2), to $\alpha = 8$ (Figure 3, left), to $\alpha = 15$ (Figure 3, right). Since the position of the "good" phenotypes in the domain is not known a priori, a nonuniform distribution of redundancy incurs the risk of under-representing them, which is not the case for a reasonably even distribution of redundancy. For this reason we take Equation (4) as defining the default CoME representation. Of course, one can obtain other similarly well-behaved

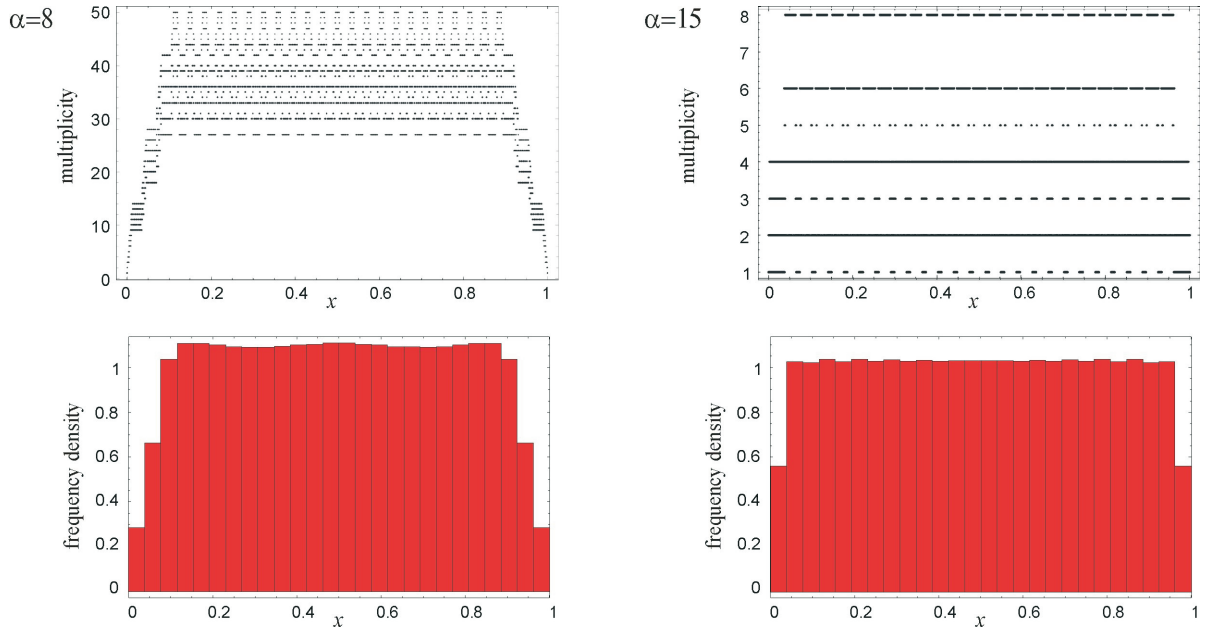

Figure 3: Using a base α that is not too small with respect to the cardinality of the alphabet the redundant positional encoding is characterized by a redundancy that is fairly uniformly distributed across the represented interval. The plots in the top row show the multiplicity of the real values obtained with Equation (4) by mapping all the strings of length $l = 4$ over an alphabet of cardinality $\beta = 26$ using a base $\alpha = 8$ (left) and $\alpha = 15$ (right). Note that the insufficient resolution of the image makes sequences of close but distinct points appear as continuous lines. The histograms in the bottom row show the frequency density of the values across the represented interval.

redundant encoding based on the general CoME mapping of Equation (1), for example using the double-base representations discussed in [4]. On the other hand, if some information about the most promising regions of the search space is available one might consider the possibility of using a definition of the weights $w(i)$ that results in a nonuniform distribution of the redundancy, or the possibility of distributing the particles nonuniformly along the interval I (Figure 1).

3. GENOME AND GENETIC OPERATORS

3.1 Mutation

Like the conventional binary and Gray-coded representations, the genotype-to-phenotype map of CoME presented in the previous section can produce different granularities of the discretization using different lengths of the strings encoding the parameters. More characters in the string correspond to more particles in the system of Figure 1 and give a finer control of the position of its center of mass. Given an evolutionary problem we do not know in general what is the resolution required to represent each parameter optimally. Moreover, the optimal resolution can be different for different parameters and can even change during the course of the evolutionary run. It is therefore useful to let the evolutionary process modify the length of the strings used to encode each parameter. To this end, in addition to the traditional genetic operator of character substitution, CoME uses two further mutation operators: one of character insertion and one of character deletion. As a consequence, strings corresponding to different parameters may have different lengths. If only numerical parameters need to be encoded, the different parameters can be kept separated for mutation and decoding. Alternatively, we can insert some markers in the genome to signal the position of the string encoding each parameter. In the following we will assume that only numerical parameters are encoded in the genome and call the string of characters associated to one parameter a *chromosome*. Therefore, the number of chromosomes corresponds to the number of parameters encoded in the genome. The presence of the deletion operator can reduce the length of a chromosome to zero. An individual with such a chromosome must be considered nonviable and removed from the population.

3.2 Crossover

In order to deal with chromosomes of variable length in CoME the crossover points have to be selected taking into account the length of the involved chromosomes. The easiest way to do this is to choose a relative position in a random chromosome. After choosing a random chromosome index d, we select the crossover points inside the corresponding chromosomes c_d^1 and c_d^2 of the two parents by drawing a random number $\phi \in [0,1]$. For each of the chromosomes we calculate the crossover point inside the chromosome as the integer nearest to $\phi\, l_c^i$, where l_c^i is the length of the d-th chromosome of the i-th parent (Figure 4). In the case where all the chromosomes have the same length this procedure corresponds to the classical crossover operator.

4. EXPERIMENTS

In order to assess the performance of CoME we consider a set of benchmark problems from function optimization. It can be argued that in the light of the no free lunch theorem [13] results gained from such benchmarks have no general validity. Nevertheless, for the practical application the results from such a benchmark

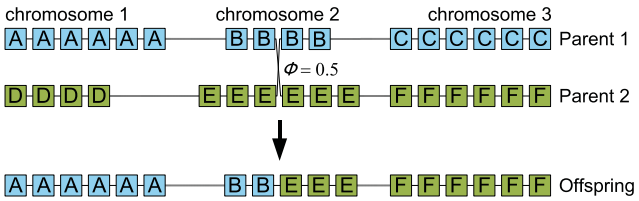

Figure 4: An illustration of the crossover operator used in the experiments reported below. It is defined so as to be applicable to two genomes composed by chromosomes of different lengths.

can be very useful if the test functions cover the typical properties of real world optimization problems. A standard generational genetic algorithm [1] with tournament selection and tournament size of 2 is used to minimize each test function within 100′000 function evaluations. The parameters of the algorithm used for the experiments are listed in Table 1. We first compare the performance of CoME to that of other standard encodings for genetic algorithms. To enlarge the scope of the comparison we consider two further algorithms. The first is the dynamic parameter encoding algorithm (DPE) [9]. DPE is a genetic algorithm that refines the mapping of the parameters by dividing the search interval into four quarters. The algorithm keeps track of the convergence of the population. If a given trigger threshold on an internal convergence measure is exceeded, the search interval is restricted by a zoom operator, which remaps the target interval and hereby doubles the precision. In the experiments described below we use a convergence threshold of 0.61, a DPE time constant of 2 and a sigma scaling factor of 2.0. The second additional algorithm is a state-of-the-art evolution strategy, the Covariance Matrix Adaptation (CMA-ES) [3]. As in [11], CMA-ES is used here with rank-μ-updates and two different population sizes, $\lambda = 200$ and $\lambda = 500$, referred to as "CMA200" and "CMA500".

4.1 Representation

For the experiments based on the genetic algorithm we consider several different methods to decode the chromosomes into discrete parameter values. As the parameter range is constrained for the test functions considered here we scale the decoded parameter values so that the obtainable values match the constraints on the parameter range. That is, given the parameter range $[x_{min}, x_{max}]$ a parameter x represented by the string $\mathbf{s}$ is scaled by

$$x'(\mathbf{s}) = x_{min} + (x_{max} - x_{min})x(\mathbf{s})$$

4.1.1 Binary Representation

Binary encoding interprets the characters c_i of a binary string as digits of a binary number. This corresponds to Equation (3) with $\beta = 2$. Each chromosome is a binary string of constant length l. We consider the cases $l = 10$ and $l = 20$, referred to as "10Bit" and "20Bit", respectively.

4.1.2 Gray Code

Gray code interprets a binary string of l bits as an integer in such a way that the representation of two neighboring numbers only differ by one bit [2, 7]. For the experiments we consider $l = 10$ and $l = 20$, referred to as "10Gray" and "20Gray".

4.1.3 Center of Mass Encoding

We use the redundant positional encoding of CoME defined by Equation (4). For the experiments we consider CoME encodings with $\beta = 26$ and different exponential bases $\alpha = 10$, $\alpha = 15$, $\alpha = 20$, and $\alpha = 26$, referred to as "CoME10", "CoME15", "CoME20", and "CoME26", respectively. Note that CoME26 has $\alpha = \beta$ and corresponds to a variable length non-redundant positional encoding. The choice of the value of cardinality of the alphabet is not critical, provided it is large enough to allow the choice of a base that results in a reasonably uniform distribution of redundancy, as explained in Section 2. In order to evaluate the effect of a nonuniform distribution of redundancy we also consider the case of $\alpha = 1$ (Equation (2)) referred to as "TOT26". Furthermore, we take into account the fixed length version of non-redundant positional encoding (Equation (3)) with $\beta = 26$ and $l = 10$ or $l = 20$, referred to as "10B26", "20B26".

Parameter	Value
Population Size	100
Mating Pool Size	99
Elite Size	1
Offspring per Parent	1
Recombination Probability	0.05
Substitution Probability	0.001
Insertion Probability (see Section 3)	0.001
Deletion Probability (see Section 3)	0.001

Table 1: The GA parameters used in the experiments; insertion and deletion are only used for the variable length genome representations. The values for substitution, insertion, and deletion probability are probabilities per character.

4.1.4 Dynamic Parameter Encoding

In the experiments described below we use a Gray code based DPE. We consider $l = 10$ and $l = 20$, referred to as "10DPE" and "20DPE".

4.2 Test Functions

To compare the performance of the different representations and algorithms we use the set of test functions for function optimization presented in [11]. Whitley *et al.* motivate their choice of using test functions with a limited domain by stating that the parameters in real world optimization problems are typically constrained. Their set contains a range of standard functions such as Rosenbrock's F2 function, the multimodal symmetric and separable Schwefel function and the non-separable and highly multimodal Rana and F101 functions. Additionally the composite F8F2 function is considered as well as two modifications of the well known Griewangk function, G1 and G2. See Table 2 for the formal definitions of the functions.[2] As in [11], the functions defined in two dimensions $F_{2D}(x,y)$ are expanded to $N = 20$ dimensions using

$$f(x_1,\ldots,x_N) = \sum_{i=1}^{N-1} F_{2D}(x_i, x_{i+1}) \qquad (5)$$

We also consider rotated instances of the test functions obtained by rotating the functions by 22.5°, translating them by 5% and scaling them by 5% in every dimension [11].[3] For each of the test functions we do 25 runs of 100'000 function evaluations with each of the aforementioned genetic representations and the two additional algorithms. The CoME chromosomes are initialized with $l = 10$.

[2]The G1 function as specified in [11] is not defined over the whole search space $\Omega = [-511, 512]^{20}$. See Table 2 for our definition.

[3]As the constraints on the parameters are applied after the rotation, the optima change. This explains why CMA-ES does not perform equally on the rotated versions even though the algorithm itself is invariant to rotation, translation and scaling.

Name	Function
Rosenbrock (F2)	$f(x,y) = 100(x^2-y)^2+(1-x)^2$
F101	$f(x,y) = -xsin(\sqrt{\lvert x-(y+47)\rvert})-(y+47)sin(\sqrt{(y+47)+x/2})$
F8F2	$f(x,y) = 1+\sum_{i=1}^{N}(\frac{F2(x,y)^2}{4000})-\prod_{i=1}^{N}cos(F2(x,y))/\sqrt{i}$
G1	$f(x_1,\ldots,x_N) = \begin{cases} \sum_{i=1}^{N}\frac{x_i^2}{4000N} - -log\left[\left[\prod_{i=1}^{N}(cos(x_i)+0.1)\right]+1\right] & \text{if } \prod_{i=1}^{N}(cos(x_i)+0.1) > -1 \\ \infty & \text{otherwise} \end{cases}$
G2	$f(x_1,\ldots,x_N) = \sum_{i=1}^{N}\frac{x_i^2}{4000N} - -1.5^{N/4}\left[\prod_{i=1}^{N}\sqrt{cos(x_i/N+i)+1}\right]^{1/4}$
Rana	$f(x,y) = xsin(\sqrt{A})cos(\sqrt{B})+(y+1)cos(\sqrt{A})sin(\sqrt{B})\ ;\ A=\lvert y+1-x\rvert\ ,\ B=\lvert x+y+1\rvert$
Schwefel	$f(x_1,\ldots,x_N) = \sum_{i=1}^{N}\left(-x_i sin(\sqrt{\lvert x_i\rvert})\right)$

Table 2: The test functions used in the benchmark experiments. The functions F2, F101, F8F2 and Rana which were originally defined for a two dimensional search space are expanded to $N = 20$ dimensions using the simple expansion method given in Equation (5). The Rosenbrock functions F2 and F8F2 are used on a domain of $[-2.048, 2.047]^N$, all other functions use $[-512, 511]^N$.

4.2.1 Adaptivity

One of the main advantages of an adaptive encoding like CoME is that it allows the algorithm to adapt the granularity of its parameters to the scaling of the problem. In order to evaluate this feature we consider two example problems with artificially scaled test functions. For 5-dimensional instances of the rotated Rana function and of the rotated F101 function we scale the first three dimensions by a constant factor. This is done by multiplying the respective parameters by 100'000 after decoding. Outside the parameter range of $[-512,511]^5$ the functions are continuously extended for each parameter by the quadratic penalty function

$$g(x_i) = \begin{cases} 10^{-5}(x_i-511)^2 & x_i > 511 \\ 10^{-5}(x_i+512)^2 & x_i < -512 \end{cases}$$

This makes it reasonably easy for the GA to find the interesting part of the search space inside of $[-512,511]^5$. On 25 runs of 100'000 function evaluations each we compare the performance of DPE10 with CoME15. CoME15 is initialized with chromosomes of length $l = 1$.

4.3 Results and Discussion

Figure 5 shows the results of the optimization runs on the different test functions. Since we can expect that different algorithms and representations are differently suited to different functions, it is not surprising that no algorithm outperforms all the others on all the functions of the test set. Nonetheless, the comparison reveals some interesting points.

In the experiments based on the genetic algorithm and a fixed-length encoding, the adoption of a 26-letter alphabet (10B26 and 20B26) produces results that are either comparable or significantly better than those obtained with a binary encoding (10Bit and 20Bit). For some functions, however, the gap between binary and 26-letter alphabet can be closed or reversed using a binary alphabet with Gray coding (10Gray and 20Gray). As expected, the totalistic encoding (TOT26) performs very poorly on several functions, presumably due to the poor match of its highly nonuniform distribution of redundancy with the properties of the search spaces determined by those functions (and, correspondingly, TOT26 performs very well on some functions, to which presumably its distribution of redundancy is well suited). The non-redundant version of CoME (CoME26), which corresponds to a variable length positional encoding, outperforms the fixed-length positional encodings that use the same alphabet (10B26 and 20B26). The interesting point is that in most cases there is a significant improvement in the transition from a fixed-length representation to the variable length representation that characterizes CoME. The effect of the transition from a non-redundant variable-length representation (CoME26) to a corresponding redundant representation with a fairly even distribution of the redundancy (CoME10, CoME15, CoME20) is less clear-cut. Although in some cases the redundancy has a beneficial effect, the reverse is true in other cases (although never with the significant decrease of performance that is observed with TOT26). Summing up, judging from the results obtained with the present test set, there seems to be a definite advantage when using a simple genetic algorithm in passing from a standard binary encoding to the variable-length representation and higher-cardinality alphabet that characterizes CoME.

The comparison of the results obtained with CoME using a simple genetic algorithm, with those obtained with the DPE algorithm (10DPE and 20DPE) and the CMA-ES algorithm (CMAES200 and CMAES500) shows that the improvement in performance due to CoME puts the genetic algorithm on the same level with these more sophisticated algorithms. As mentioned above, no combination of algorithm and representation outperforms all the others, and combinations that produce excellent results on some function (e.g., CMA-ES on Rosenbrock's and G1 and G2 functions, or DPE on the unrotated Rosenbrock's function) perform poorly on other functions (e.g., CMA-ES on the F101 and Rana function, or DPE on the G1 function). From this point of view, the most interesting observation stemming from the results illustrated in Figure 5 is that CoME maintains a reasonable performance on all the functions of the test set and never shows the dramatic decrease in performance manifested by DPE and CMA-ES on some functions.

Figure 6 shows the results of the experiments on adaptivity described in Section 4.2.1. In these experiments the first three coordinate axes of the 5-dimensional domain are scaled so as to require a finer resolution of the discretization for the correct location of the minima. The plots in the left column of Figure 6 show that there is no significant difference in the best function values obtained with CoME and DPE. In both experiments the CoME chromosomes encoding the scaled coordinates show a significant increase in length relatively to the chromosomes encoding the two unscaled coordinates (Figure 6, right column). Using a base $\alpha = 15$ for the exponential in Equation (4), each additional character in the string multiplies the number of available parameter values by about 15. An increase of a factor $100'000$ in the resolution would suggest the need for an additional five characters ($log(100'000)/log(15) \approx 4.25$).

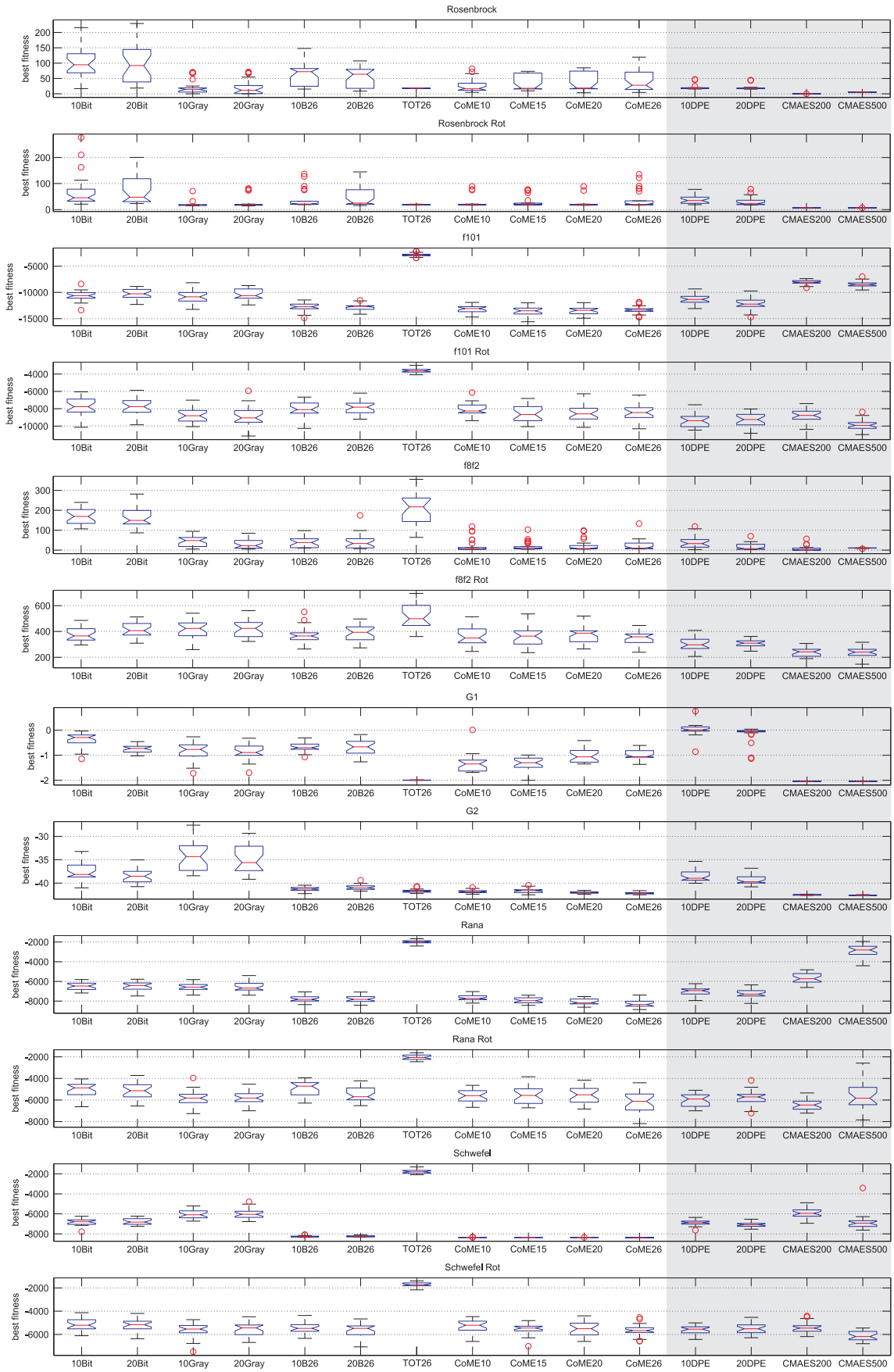
Rosenbrock
Rosenbrock Rot
f101
f101 Rot
f8f2
f8f2 Rot
G1
G2
Rana
Rana Rot
Schwefel
Schwefel Rot
best fitness
10Bit
20Bit
10Gray
20Gray
10B26
20B26
TOT26
CoME10
CoME15
CoME20
CoME26
10DPE
20DPE
CMAES200
CMAES500

Figure 5: *(previous page)* Results from 25 minimization runs of 100'000 function evaluations each. The results shown on a white background were obtained with the genetic algorithm, those shown on a gray background were obtained with the DPE and CMA-ES algorithms. The midline in each box is the median, the borders of the box represent the upper and the lower quartile. The whiskers outside the box represent the minimum and maximum values obtained, except when there are outliers which are shown as small circles. We define outliers as data points which differ more than 1.5 times the interquartile range from the border of the box. The notches permit the assessment of the significance of the differences of the medians. When the notches of two boxes do not overlap, the corresponding medians are significantly different at (approximately) the 95% confidence level [6].

This matches well the results observed in our CoME15 runs, where the average chromosome lengths at the end of the runs display a difference of 4.86 and 5.73 characters between the chromosomes that encode the scaled and nonscaled axes of the domains of the two test functions. This shows that CoME is capable of self-adapting the granularity of the genetic representation independently for each parameter according to the needs of the problem at hand.

5. CONCLUSION

We have introduced the Center of Mass Encoding (CoME), a new class of genetic representations of real-valued parameters that allows self-adaptation of the resolution of the representation and permits the choice of the degree of redundancy of the representation. Using an extensive set of test functions as a benchmark we have compared the performance of a genetic algorithm using the CoME representation with that of other widely used representations such as binary encoding and Gray code encoding, and with two more sophisticated adaptive evolutionary algorithms, namely, Dynamic Parameter Encoding (DPE) and Covariance Matrix Adaptation (CMA-ES). Not only did a genetic algorithm using CoME perform very well compared to the same algorithm using the conventional discrete representations, but it outperformed the more advanced evolutionary algorithms on some of the test functions, while displaying a consistently good behavior on the whole test set. Furthermore, we have presented the results of two experiments that illustrate how CoME is able to self-adapt the resolution of the representation to the requirements of the evolutionary problem. In summary, we have shown that CoME is a very powerful new class of encodings for the representation of real-valued parameters in evolutionary algorithms.

Acknowledgments

The implementation of CMA-ES used here was adapted from original code by Nikolaus Hansen. The implementation of DPE was adapted from original code by Nicol Schraudolph. The implementation of the test functions was taken from the webpage of the GENITOR group at Colorado State University. Many thanks to Daniel Marbach and Andrea Soltoggio for discussions and comments on the manuscript. This work was supported by the Swiss National Science Foundation grant no. 200021-112060.

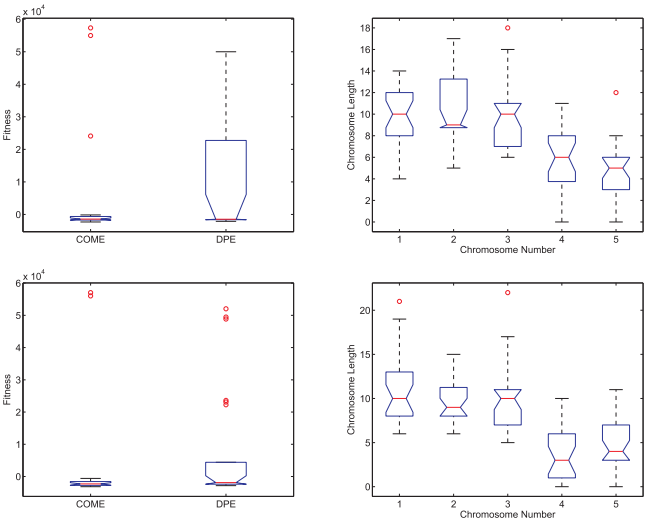

Figure 6: The two graphs on the left show the minimal function values obtained by CoME15 and DPE10 in 25 runs after 100'000 evaluations on the scaled versions of the 5-dimensional rotated Rana function (top row) and rotated F101 function (bottom row) described in Section 4.2.1. The two graphs on the right show the distribution of the chromosome lengths of the solutions found by CoME15 (DPE uses fixed length chromosomes of length 10). For the details of the boxplot format see the caption of Figure 5

6. REFERENCES

[1] T. Bäck, D. Fogel, and Z. Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics, Bristol, 2000.

[2] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[3] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996.

[4] Y.-D. Kwon, S.-B. Kwon, S.-B. Jin, and J.-Y. Kim. Convergence enhanced genetic algorithm with successive zooming method for solving continuous optimization problems. *Computers & Structures*, 81(17):1715–1725, Aug. 2003.

[5] C. Mattiussi. *Evolutionary synthesis of analog networks*. PhD thesis, EPFL, Lausanne, 2005.

[6] R. McGill, J. W. Tukey, and W. A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, Feb. 1978.

[7] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 2nd edition, 1992.

[8] F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Springer, Berlin, 2006.

[9] N. Schraudolph and R. Belew. Dynamic Parameter Encoding for genetic algorithms. *Machine Learning*, 9(1):9–21, 1992.

[10] R. Streifel, I. Marks, R.J., R. Reed, J. Choi, and M. Healy. Dynamic fuzzy control of genetic algorithm parameter coding. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 29(3):426–433, 1999.

[11] D. Whitley, M. Lunacek, and A. Sokolov. Comparing the niches of CMA-ES, CHC and Pattern Search using diverse benchmarks. In T. R. et al., editor, *Proceedings of PPSN IX*, volume 4193 of *LNCS*, pages 988–997, Berliin, 2006. Springer.

[12] D. Whitley, K. Mathias, and P. Fitzhorn. Delta Coding: An iterative search strategy for genetic algorithms,. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 77–84, San Mateo, CA, 1991. Morgan Kaufman.

[13] D. Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 2001.

[14] A. Wu and I. Garibay. The proportional genetic algorithm: Gene expression in a genetic algorithm. *Genetic Programming and Evolvable Machines*, 3(2):157–192, 2002.