Elitist Generational Genetic Chromodynamics - a New Radii-Based Evolutionary Algorithm for Multimodal Optimization

Catalin Stoean
University of Craiova
Faculty of Mathematics and Computer Science
Department of Computer Science
Str. Alexandru Ioan Cuza, No. 13
Craiova, 200585, Romania
catalin.stoean@inf.ucv.ro

Mike Preuss
University of Dortmund
Department of Computer Science
Chair of Algorithm Engineering and Systems Analysis
Joseph-von-Fraunhofer-Str. 20
44227, Dortmund, Germany
mike.preuss@uni-dortmund.de

Ruxandra Gorunescu
University of Craiova
Faculty of Mathematics and Computer Science
Department of Computer Science
Str. Alexandru Ioan Cuza, No. 13
Craiova, 200585, Romania
ruxandra.gorunescu@inf.ucv.ro

D. Dumitrescu
Babes - Bolyai University
Faculty of Mathematics and Computer Science
Department of Computer Science
Str. Kogalniceanu, No. 1
Cluj-Napoca, 400084, Romania
ddumitr@cs.ubbcluj.ro

Abstract- A new radii-based evolutionary algorithm (EA) designed for multimodal optimization problems is proposed. The approach can be placed within the genetic chromodynamics framework and related to other EAs with local interaction, e.g. using species formation or clearing procedures. The underlying motivation for modifying the original algorithm was to preserve its ability to search for many optima in parallel while increasing convergence speed, especially for complex problems, through generational selection and different replacement schemes. The algorithm is applied to function optimization and classification; obtained experimental results, in part improved immensely by state-of-the-art parameter tuning (SPO), encourage further investigation.

1 Introduction

Belonging to the family of radii-based multimodal evolutionary frameworks, Genetic Chromodynamics (GC) [1] has recently proven to be a very powerful tool [2], [3], [4]. GC builds and maintains subpopulations, corresponding each to a global/local optimum of the problem, through the use of a stepping stone search mechanism and a local interaction principle, as selection for reproduction. Selection for replacement takes place between resulting offspring and the current "stone" chromosome. A merging operator is used to achieve decrease in the number of individuals.

The work presented here goes further in the development of another evolutionary algorithm in this framework. The selection for replacement strategy used in our new algorithm is generational. In contrast to the stepping stone mechanism, the first parent is selected randomly. The offspring obtained after crossover does not replace any of the parents particularly, but the worst chromosome (with respect to fitness values) within a replacement radius, a new parameter of the algorithm. The local interaction principle and merging still hold.

The reason for creating the new algorithm, Elitist Generational GC (EGGC), was that of preserving the ability of GC to properly locate several or all optima within one go, and additionally speed up this process. The new algorithm always achieves the first goal, the second one being accomplished for more complex problems (for instance, for higher values for n in n-dimensional problems), due to changes that lead to better search space exploitation.

The algorithm is tested on several largely used benchmark functions, i.e., Six-Hump Camel Back, Schaffer, Himmelblau and Schwefel and on classification of medical data.

The paper is organized as follows: the next section presents an overview of the GC framework; section 3 describes the modified algorithm; section 4 places GC in the context of related evolutionary techniques. Futhermore, section 5 reports experimental results obtained for optimization of several multimodal test functions and for classifying medical data, respectively. These are partly backed up by a recent parameter tuning method. Finally, we conclude with a discussion and an outlook onto possible future work.

2 Genetic Chromodynamics framework

The GC framework has demonstrated success in application to function optimization, clustering and classification. In particular, it is able to concentrate search on many basins of attraction in parallel, so that several optima are found simultaneously. The evolutionary process takes place as follows.

First, the initial population is randomly generated. Next, every chromosome is taken into account in the forming of the new generation. Mating regions around each chromosome are determined by a radius. Therefore, only neighboring chromosomes are recombined. When no mate is found in the mating region of the current chromosome, the latter produces one offspring by mutation, with a step size that still keeps the descendant in the mating region of its par-

ent. If there is more than one chromosome *near* the current, the mate is determined using proportional selection. Then, if the offspring has better fitness than the current chromosome, it replaces the latter in the population.

Algorithm 1 Merging procedure

repeat
  a chromosome c is considered the current chromosome;
  select all m chromosomes in the merging region of c, including itself;
  remove all but the best chromosome from the selection;
until merging can not be applied at all

At every iteration, each chromosome is important for building the new generation. By reducing potential partners for crossover of a chromosome to those lying in its mating region, only chromosomes that are close to each other recombine, favoring the appearance and maintenance of subpopulations. Offspring replaces the current chromosome only if fitter. Thus, after a few generations, the chromosomes will concentrate on the most promising regions of the search space, $i.e.$ those connected to the optima. Moreover, selection for replacement is based on a quasi-generational strategy since, if replacement takes place, the mating group of another certain chromosome in the current generation changes, be that the replaced parent belonged to it.

Algorithm 2 GC algorithm

t = 0;
initialize population P(t);
repeat
  evaluate each chromosome;
  for all chromosomes c in the population **do**
    if mating region of c is empty **then**
      apply mutation to c;
      if obtained chromosome is fitter than c **then**
        replace c;
      end if
    else
      choose one chromosome from the mating region of c for crossover by proportional selection;
      obtain and evaluate one offspring;
      if the offspring has better fitness than c **then**
        replace c
      end if
    end if
  end for
  merging;
  $t = t + 1$;
until stop condition

{Remark: Usually, the stop condition refers to the total number of generations or to the number of generations passed without any major improvement of the solutions.}

For the subpopulations to become better and better separated with each iteration, GC reduces the population size in a way that preserves the position of each subpopulation. In this respect, a new operator, called merging, is introduced. It merges very similar chromosomes and is applied after perturbation takes place. Let a chromosome c be given. If the distance between c and another chromosome is very small, under a given radius, then the latter is considered part of the merging region of c. From the set of all chromosomes in the merging region of c only one is kept. Generally, it is chosen such that the one with the best fitness evaluation is kept in the population. Alternatively, other merging schemes may be used (for instance, the mean of the chromosomes in the merging region). The merging procedure is outlined in algorithm 1. The interplay between merging and mating regions strongly impacts convergence properties: a large merging radius leads to deletion of potential mates of the surviving individuals and thus blocks crossover. Therefore, it is usually chosen much smaller than the mating area. Along with separation, the worse chromosomes are removed step by step, and the process gradually transforms from a globally–oriented population–based to a parallel local search algorithm. Thus, in the end, only one chromosome remains connected to any optimum. Each such chromosome now corresponds to a single hillclimber that only uses mutation as variation operator. Figure 1 illustrates radii-based mating and merging, while algorithm 2 outlines high-level pseudocode.

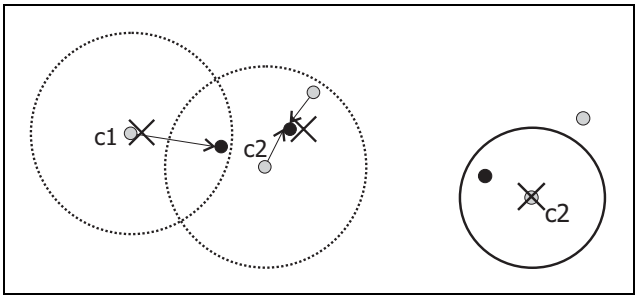

Figure 1: *Mating (left) and merging (right) within GC. Chromosome c1 produces one offspring by mutation, chromosome c2 selects another chromosome from its mating region (dotted circle) and produces one offspring by crossover. Crossed lines indicate replaced chromosomes with worse fitness. During merging, c2 is deleted because there is another chromosome with better fitness in its merging region (solid circle).*

3 Elitist Generational GC algorithm

In our new algorithm, the original scheme was modified in order to achieve increased convergence speed based on better exploitation of the search space. That is obtained especially by the way in which offspring resulting from crossover enters the population. It does not replace the first parent, but the worst chromosome in its replacement radius. Therefore, weak chromosomes are removed more aggressively (alongside with the effect merging has in this respect) from the current population. For this reason, the stepping stone principle is not applied here, but n (where n is the number of chromosomes in the population) random chromosomes are selected instead. Now, chromosomes may be replaced by some offspring without ever being selected for recombination. However, an important aspect of the al-

gorithm is the choice of the replacement radius value. If picked properly, this new parameter may lead to improved convergence speed.

Algorithm 3 EGGC algorithm

```
t = 0;
initialize population P(t);
repeat
  evaluate each chromosome;
  for i = 1 to n do
    randomly choose a chromosome c;
    if mating region of c is empty then
      apply mutation to c;
      if obtained chromosome is fitter than c then
        replace c;
      end if
    else
      choose one chromosome from the mating region of c for
      crossover by proportional selection;
      obtain and evaluate one offspring d;
      find worst chromosome w within replacement radius of
      d;
      if d has better fitness than w then
        replace w;
      end if
    end if
  end for
  merging is applied to all chromosomes;
  t= t + 1;
until stop condition

{Remark: The offspring d obtained after crossover can replace
a chromosome that does not belong to any of the mating regions
of its parents, thus performing faster separation of chromosomes
into clusters.}
```

Selection for replacement adopts a generational scheme, as already stated. As the effect of quasi-generational selection for replacement, used by the classical GC, can be noticed only when another stepping-stone does not find the initial chromosome in its mating region but sees the offspring instead, the new algorithm is totally generational. This means that the offspring that replaces its parent might be selected for reproduction many times in the same generation or might vanish within that generation. Thus, the generational scheme leads to increased exploitation.

The local interaction, crossover, mutation and merging principles still hold. Radii-based evolution in the new context of EGGD is depicted in figure 2 and the complete algorithm is outlined in alg. 3.

4 GC and related evolutionary algorithms

Both concrete algorithms from the Genetic Chromodynamics framework presented here are radius-based methods for multimodal optimization. Thus, they are related to several other evolutionary algorithm types that use properties of the search space distribution of a population to maintain several search paths at the same time. In the ideal case, every search path corresponds to one basin of attraction (and, if the number of basins is small, also vice versa), so that every

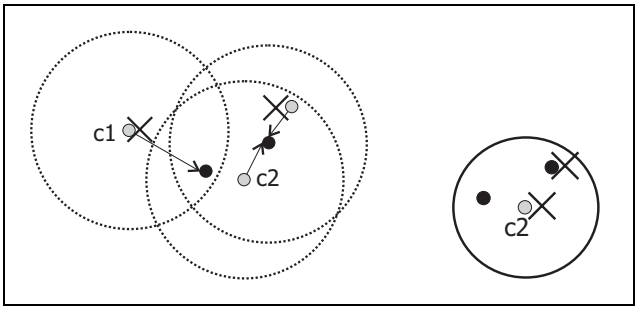

Figure 2: *Mating (left) and merging (right) within EGGC. As in figure 1, c1 and c2 each produce one offspring. This time, the second offspring replaces its other parent because the latter is the worst chromosome in its replacement region. During merging, two chromosomes are removed, c2 and one offspring, because now three chromosomes are within merging radius from c2.*

run results in finding many optima, be they local or global. Such information may be valuable by itself if multiple solutions are desired; additionaly, it helps in determining the global optimum of a problem.

A first similar approach we know of is the Species Conservation algorithm [5] where a radius around each individual is used to determine groups of individuals (species) that represent similar solutions. A specialized selection operator guarantees survival of at least the best individual of each group. GC algorithms achieve the same implicitly by local selection and merging, without ever detecting a species as a whole. This does not necessarily make them faster, but in our opinion the GC concept is simpler.

A second approach, even more similar to that of GC, is the Clearing procedure [6]. It is a niching algorithm where sharing of resources within every subpopulation takes place only between the winners, the fittest of that niche, instead of applying to all its individuals as standard. The number of winners is generally set to 1, but their number, called capacity of a niche, can be higher. While GC performs crossover only between individuals belonging to the same mating region, clearing performs global recombination. GC sets two different radii for the identification and maintenance of niches in the population, one for mating and one for merging, while the other algorithm utilizes a single radius, for the clearing region. The approaches further differ in the effect of merging: the Clearing procedure decreases population size only temporary to circumvent selection of dominated individuals for reproduction.

The use of mating restrictions is another well-known feature of EAs for multimodal optimization, in island or diffusion (or, more generally, topology-based) models as well as for explicit niching algorithms [7], [8], [9], [10]. In GC, mating as well as merging radius is measured in search space, and it is usually chosen such that some potential partners survive merging, thereby enabling recombination in the next generation.

On the other hand, merging in GC usually leads to decreasing population sizes. In case the radii parameters have been set to proper values, depending on the problem,

subpopulations crowding one basin of attraction each get smaller and smaller until only one individual — per optimum — is left that performs a simple hillclimbing strategy, similar to a $(1 + 1)$ evolution strategy [11]. In reality, it is often difficult to find these parameter values — or even impossible, when the sizes of basins of attraction have large variance. However, we expect that much yet unused potential is hidded here that can be exploited with modified GC variants of which EGGC is a first attempt.

5 Numeric optimization experiments

The proposed algorithm was tested on three bi-dimensional functions and an n-dimensional one (table 1) and obtained results indicate that it has good accuracy, stability, low computational time and thus provides a good method for multimodal function optimization.

5.1 Function optimization

Table 1: *The four considered multimodal test problems*

- Six-Hump Camel Back (2D)
  $f_1(x, y)$ = -[(4 - 2.1x^2+$\frac{x^4}{3}$)x^2 + xy + 4(y^2 - 1)y^2]
  where -2 $\leq x \leq$ 2 and -2 $\leq y \leq$ 2
- Schaffer function (2D)
  $f_2(x, y)$ = -0.5-$\frac{sin^2(\sqrt{x^2+y^2})-0.5}{(1+0.001(x^2+y^2))^2}$
  where -20 $\leq x \leq$ 20 and -20 $\leq y \leq$ 20
- Himmelblau function (2D)
  $f_3(x, y)$ = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2
  where -4 $\leq x \leq$ 4 and -6 $\leq y \leq$ 6
- Schwefel function (100D)
  $f_4(\overline{x})$ = $\sum_{i=1}^{n} x_i sin\sqrt{|x_i|}$
  where -500 $\leq x_i \leq$ 500 and $i \in \{1, 2, ..., n\}$

The aim for the *Six-Hump Camel Back* function is to locate the two global optima and the four other local optima. The main issue here is that the two local optima with the same value -2.1043 can easily be missed by a multimodal evolutionary algorithm. For instance, in [12], several algorithms were used (a genetic algorithm using gradient information, a local optimization method, a multi-start local optimization method and a conventional GA), but each time only one local or global optimum was detected with an accuracy of $\epsilon = 10^{-3}$. Because of lack of information, direct comparisons to GC algorithms could not be conducted. In table 2, results obtained by the original GC algorithm, EGGC and two algorithms presented in [13] are outlined. The values for the EGGC parameters used for f_1 are shown in table 7. The success rate was computed as the ratio between the number of cases when all optima have been located with the desired accuracy and the total number of runs. For the two algorithms from [13], local optima accuracies had not been specified. Although the number of generations in EGGC was lower than that in the classical algorithm of GC, the number of evaluations in EGGC was a little higher; the explanation is that after each crossover, there are some more evaluations that are performed for all chromosomes in the replacement area of the offspring. To conclude, the two algorithms perform in a very similar manner for this function.

Table 2: *Comparisons for the Six-Hump Camel Back function*

Measures	CDE and MCDE	MMDE	GC	EGGC
No. of runs	30	30	30	30
Acc. global opt.	10^{-5}	10^{-5}	10^{-5}	10^{-5}
Acc. local opt.	–	–	10^{-4}	10^{-4}
Success rate	100%	100%	100%	100%
Mean eval calls	62 645	14 610	19 923	19 980

The aim for the *Schaffer* function is to detect the global optimum $f_2(x, y)$ = 0 that can be escaped because of the high number of local optima around it and because the difference between the values of the local optima and the value of the global optimum is very small (of order 10^{-3}). This is the reason why the parameters for f_2 (table 7) were chosen such that the entire search space was covered. As noticeable from table 3, the original GC algorithm, although performing well, needs many more evaluations than EGGC.

Table 3: *Performance of GC and EGGC on the Schaffer function*

Results	GC	EGGC
No. of runs	30	30
Accuracy global optimum	10^{-6}	10^{-6}
Success rate	100%	100%
Mean eval calls	658 211	349 712

The *Himmelblau* function has four global optima, $f_3(x, y)$ = 200. In [14], it was tested using a sequential niche technique; again, obtained results cannot be directly compared with those obtained by GC/EGGC due to the fact that in [14], to detect all the four optima, several runs (6.1, on average) of the algorithm were necessary. Additionaly, the desired accuracy had not been specified. Using the proposed algorithm, all four optima were detected for each of the 30 runs with the accuracy ϵ = 10^{-5}. A performance comparison between GC and EGGC is given in table 4.

Table 4: *Performance of GC and EGGC on Himmelblau's function*

Measures	GC	EGGC
No. of runs	30	30
Accuracy global optima	10^{-4}	10^{-4}
Success rate	100%	100%
Mean eval calls	87 307	97 817

The aim for the Schwefel function is to find the global optimum $f_4(\overline{x})$ = 418.9829n. In this paper, f_4 was considered in turn for $n = 1, 2, ..., 100$. Although radii-based evolutionary algorithms usually have difficulties with high-dimensional test problems, the proposed algorithm detects the optimum with an accuracy of $\epsilon = 10^{-2}$, even for n = 100, in all 100 runs, but only after a high number of fitness evaluations. The accuracy improves with the decrease of the number of dimensions.

The Schwefel function was also tested with the original GC algorithm which detected the optimum for n = 100 with similar accuracy, but needed even more fitness evaluations than EGGC. We present a comparison of both algorithms and different performance measures for n = 10 and $\epsilon = 10^{-2}$ in table 5.

Table 5: *GC and EGGC performance on the Schwefel function when n=10*

Numerical results	GC	EGGC
No. of runs	30	30
Average evaluations to solution	1 819 593	1 545 752
Mean generations	48 747	39 043
Mean best fitness	4189.827822	4189.827956

Results for the Schwefel function were compared to those in [15]. In the latter, authors presented the behavior of four global evolutionary algorithms, four island algorithms and one cellular algorithm on the shifted Schwefel function with $n = 1, 2, ..., 10$. In the first category, a simple genetic algorithm (SGA), an elitist SGA (ESGA), the parallel CHC (pCHC) and the Genitor algorithm were present. The second category contained island-SGA (I-SGA), island elitist SGA (I-ESGA) island-pCHC (I-pCHC) and island-Genitor (I-Genitor). The algorithms were run for a set number of generations and the number of runs (out of 30) in which the global optimum was found (denoted by ns) was reported, along with the mean best fitness of the 30 runs (denoted by MBF in table 6). Note that fitness values are normalized so that the global optimum is 0.0.

Table 6: *Performance of different evolutionary techniques on the Schwefel function, from [15]*

Algorithm	Ns	f_4 MBF
SGA	0	17.4
ESGA	16	17.3
pCHC	15	5.9
Genitor	20	13.2
I-SGA	9	6.5
I-ESGA	13	2.6
I-pCHC	28	0.2
I-Genitor	24	0.9
Cellular	26	0.7

Table 7 contains the empirically determined EGGC parameter values for the four functions. Mutation with normal perturbation was used in all cases; the value of the mutation strength directly depends on the size of the interval and implicitly on search space size. In case of the Schwefel function, a higher mutation strength was chosen in the beginning, to escape local optima. The value was afterwards decreased by n/100 every 100 generations, where n represents the number of dimensions. The values for mating and merging radii were chosen in the same manner, thus depending on the search space size. Replacement radius for a given chromosome is generally chosen to be equal to the mating value of that individual. However, for problems with large plateaus in the fitness landscape, by choosing a different value for the replacement from that of the mating radius, faster convergence can be achieved (see tables 5 and 7 vs. table 8 for f_4). On the other hand, for problems with very close optima, its value must be chosen again not equal to the mating radius (see f_1 in table 7), in order to obtain all optima. The last parameter from table 7 gives the number of generations without any improvement which is necessary to determine termination of the algorithm. For the Schaffer and Schwefel functions, the stop condition was established to be the moment when the algorithm achieves the desired accuracy because only the global optimum had to be determined.

Table 7: *Parameters of the algorithm for all functions*

Parameters	f_1	f_2	f_3	f_4
No. of chromosomes	100	2000	150	500
Mutation strength	0.1	0.1	0.5	$25n$
Mating radius	0.14	0.1	2	$15n$
Replacement radius	0.1	0.1	2	$15n$
Merging radius	0.14	0.1	0.02	$12n$
Mutation probability	0.4	0.4	0.4	0.4
No improvement times	100	–	50	–

5.2 Parameter tuning-supported comparison

To strengthen our assumption that for the tested high dimensional problem, EGGC is really faster than the original GC algorithm, we applied a recent parameter tuning method [16], [17]. SPO builds on a quadratic regression model, supported by latin hypercube sampling (LHS) and noise reduction by incrementally increased repetition of runs.

At first, we tuned parameters for the 10 dimensional Schwefel function. SPO was run for 6 steps with an initial design size of 50, adding up to a total of around 850 runs (table 8). The response value Y (quality criterion) was set to the average number of evaluations (AES) until reaching the global optimum with accuracy 10^{-2}.

Results show that for both algorithms, the AES decreased considerably, to $\approx 10\%$ of the original values reported in table 5. A t-test confirms that the true means for GC and EGGC are different with 95% confidence (p-value 0.036), so that we conclude that EGGC is indeed faster on this problem. However, the difference is quite small —

Table 8: *Algorithm design for optimizing the 10-dimensional Schwefel function with GC and EGGC. The two last columns give the best found configurations, the last lines the resulting AES measures with standard deviations.*

Parameter name	$\mathbb{N}/\mathbb{R}$	Min	Max	GC	EGGC
No of chromosomes	$\mathbb{N}$	10	500	10	13
Mating radius	$\mathbb{R}_+$	0.01	5.0	3.6752	2.9366
Replacement radius	$\mathbb{R}_+$	0.01	5.0	–	4.748
Merging radius	$\mathbb{R}_+$	0.01	5.0	1.275	3.9197
Mutation strength	$\mathbb{R}_+$	1.0	250.0	238.45	231.08
Mutation prob.	$\mathbb{R}_+$	0.0	1.0	0.6185	0.8575
Mutation scaledown	$\mathbb{N}$	10	1000	368	625
No improv. times	$\mathbb{N}$	1	200	167	46
AES, 32 runs	–	–	–	1.69E+5	1.45E+5
AES standard dev.	–	–	–	5.92E+4	3.63E+4

Table 9: *Algorithm design for optimizing the 20 dimensional Schwefel function with GC and EGGC; the maximum radii are increased compared to table 8.*

Parameter name	$\mathbb{N}/\mathbb{R}$	Min	Max	GC	EGGC
No of chromosomes	$\mathbb{N}$	10	500	24	10
Mating radius	$\mathbb{R}_+$	0.01	15.0	11.695	11.02
Replacement radius	$\mathbb{R}_+$	0.01	15.0	–	9.6561
Merging radius	$\mathbb{R}_+$	0.01	15.0	14.768	3.81
Mutation strength	$\mathbb{R}_+$	1.0	250.0	242.16	237.43
Mutation prob.	$\mathbb{R}_+$	0.0	1.0	0.6705	0.6185
Mutation scaledown	$\mathbb{N}$	10	1000	269	368
No improv. times	$\mathbb{N}$	1	200	8	167
AES, 32 runs	–	–	–	3.32E+5	1.07E+5
AES standard dev.	–	–	–	1.10E+5	2.26E+4

around 15% — and may be statistically significant, but of little importance for practical uses. Surprisingly, many optimized parameter values are very similar for both algorithms, e.g. number of chromosomes, mating radius, and the mutation strength and probability. In contrast to that, the merging radii are chosen differently; larger than the mating radius for EGGC, and smaller for GC. Nevertheless, other good configurations found during tuning indicate that smaller merging radii also work for EGGC.

From these first results, and supported by our previous findings, we learned that EGGC may be advantageous for higher dimensional problems. Consequently, we performed a second comparison for the same test function in 20 dimensions, this time allowing for larger radius values. The outcome (table 9) unambiguously favours EGGC over GC, thereby validating our assumption. However, the maximum radii tested may still be too small; increasing them may lead to even better speedup.

6 Application to a decision problem

The EGGC algorithm was applied to a decision problem (classification of medical data) for two reasons: first, to prove its suitability as general multimodal optimization technique, and second, because results can be compared to the ones reported for entirely different algorithms, namely artificial neural networks. Applications of other evolutionary algorithms to this problem have been found only in hybridization also with neural networks.

6.1 Diabetes diagnosis

EGGC will thus further address the optimization task of achieving the classification of patients, based on attributes depicted from a series of medical tests and non-medical data, into diabetes positive or negative. The data set comes from the UCI repository of machine learning databases [18]. The decision to be reached is whether a Pima indian is to be diagnosed with diabetes or not. There are eight continuous attributes containing personal data, $e.g.$, age, number of pregnancies, and medical data, $e.g.$, blood pressure, body mass index, glucose tolerance, and a last discrete attribute, the diagnosis, either 0 (negative) or 1 (positive). 34.9% are assigned diabetes positive. The total number of cases is 768.

Table 10: *Results of comparable techniques for the diabetes task*

Algorithm	Repeats	Accuracy (%)
EGGC with cross-validation	100	75.06
EGGC with Prechelt's rules	100	69.672
EGGC with random cross-validation	100	69.515
EGGC best accuracy (4 rules)	100	80
Neural Network with Prechelt's rules	100	65.5
Evolved NN with cross-validation	30	77.6
Evolved NN best result	30	80.7

The data is complete, except for some 0 values of attributes that were not reported as missing data, but look, however, strange. No replacement or deletion of these values was undertaken in proposed algorithm.

A chromosome $c = (c_1, ..., c_8, c_9)$ is a string where each of the first eight genes corresponds to an attribute of a patient. The last gene represents the outcome. Thus, a chromosome encodes an if-then rule. The condition is a conjunction of personal data and symptoms and the conclusion is the diagnosis. The fitness of a chromosome is computed as its distance to all patients of the training set that have the same outcome. Minimizing distances leads to good rules (representing cluster centers in the eight-attribute variable space) for that diagnosis. The match is determined using a distance measure between the chromosome (rule) and the patient, defined as:

$$d(u, v) = \sum_{i=1}^{n} \frac{|u_i - v_i|}{b_i - a_i} \quad , \tag{1}$$

where a_i and b_i are the lower and upper bounds, respectively, of the i-th attribute. As the values for the eight attributes belong to different intervals, the distance measure has to refer to the interval bounds. We used convex crossover, with the coefficients biased by the fitness of the two parents involved. Mutation is with normal perturbation.

At termination of the algorithm, two chromosomes are obtained, each for a class. These chromosomes are tested against the data in the test set and accuracy is computed. The ratio between training and test sets was set to 75%-25%, as established by Prechelt in [18] with respect to the diabetes task.

Three kind of tests were conducted with different possibilities of choosing the data that would go into training and test, respectively. The two sets are obviously disjoint. First, cross-validation was performed. The first 75% of the data were taken to compose the training set and the remaining data were assigned to the test set, as in the standard manner of using this data set, according to [18]. The obtained mean accuracy for the test set in 100 runs was 75.06%. Second, another test was done according to rules of splitting that should be used for this data set, as established by Prechelt in [18]. The data set is sequentially split into 75% training - 25% test to give 4 different combinations of these two sets, *i.e.* first 75% data for training – last 25% for test, first 25% data for test – last 75% for training, first 50% data for training – next 25% for test – last 25% for training again, first 25% data for training – next 25% for test – last 50% for training again. The algorithm is subject to 100 trials again.

The mean accuracy obtained for the test set was 69.672%. Last, random cross-validation was performed, $i.e.$ the training set containing 75% data and the test set containing 25% data were randomly generated in each run. The algorithm was applied 100 times and the obtained mean accuracy was of 69.515%. However, in many tests, it was noticed that when the chromosome pool still has four chromosomes left and has not converged yet, a higher accuracy of 80% is obtained. This leads to the idea that in the structure of each of the two obvious clusters two other subclusters are included. Thus, with the best instead of last accuracy, better results can be obtained. Note that applying SPO parameter tuning here did not lead to any improvement.

6.2 Result comparison

Literature reports accuracy on the diabetes task ranging from 62% to 80.7%. Unfortunately, not many papers specify variables of the testing environment; thus, proposed algorithm cannot be objectively compared with them. Yet, there are some papers which specify them although they differ in training/test sets sizes and method of assigning data to each of them. Authors also mention they did not delete any of the tuples containing missing data. When not specified, the number of runs is presumed to be ten.

In [19] a neural network algorithm to forecast the onset of diabetes mellitus was used. From the 768 samples, an equal number of 170 samples were selected randomly to represent each of the two possible results of diabetes test: positive and negative. The remaining 428 were used as validating samples. The mean of five runs was 75.12%. In [20], a total of 30% of the records were randomly selected as test set. Rules were mined from the remaining 70% of the data. The algorithm was applied ten times. If the authors were to define a baseline accuracy to mean the accuracy obtained by simply assigning the most frequently occurring values to the attributes being predicted, it is 65.1%.

One approach our results can be directly compared to is [21]. Using a neural network, 75% training - 25% test, the rules established by Prechelt, 100 trials and no replacement or deletion of missing data, the mean accuracy was obtained as 65.55%. Another approach that allows for objective comparison of results is [22]. A new evolutionary system to evolve artificial neural networks was proposed, cross-validation was used and 30 runs of the algorithm were conducted and again no replacement or deletion of missing data was done. The obtained mean accuracy on the test set was of 77.6%, the best accuracy 80.7%.

In a brief conclusion, table 10 contains a comparison between the accuracy obtained by EGGC versus those of the two algorithms. The original GC algorithm was not applied.

7 Discussion and Conclusions

The present paper describes a new algorithm in the GC framework. Through its increased exploitative nature, it seems that a better equilibrium between exploration and exploitation than in the standard GC algorithm is established. This leads to two advantages in using Elitist Generational GC.

First, it proves to be very accurate and stable in very hard multimodal cases. A first situation is that of the difficulty in locating the local optima, e.g., the Six-Hump Camel Back function. Another hard situation would be very distant optima, e.g. the Schwefel function, where other algorithms often miss the global optimum. Another situation is having more global optima which are not easily distinguishable in the fitness landscape, like in the case of the Himmelblau function. An even harder situation is that of local optima lying too close to the global optimum - thus an evolutionary algorithm usually gets stuck in a local optimum. This is the case with the Schaffer function. Some of the evolutionary algorithms fail in some of these cases, some fail in other cases. The classical GC algorithm, on the other hand, also performs well in this respect, but its speed of convergence is not very good for functions with many local optima very near to the global optimum, e.g. the Schaffer function and, especially in the case of more complex problems (n-dimensional problems), e.g. the Schwefel function.

As the question of computational time arises, for many low-dimensional multimodal functions, both algorithms are surely not competitive to other, namely non-evolutionary algorithms. For the other cases, i.e. local optima and global optima almost indistinguishable and n-dimensional function optimization with high values for n, for instance, both EGGC and GC perform accurate and stable, however, EGGC is significantly faster.

Another important feature of proposed algorithm is the new parameter, the replacement radius. Together with the generational scheme, it seems to have brought more power to EGGC. For instance, for the Schwefel function, where the optima are far from each other, this parameter promotes rapid movement of the chromosomes through the search space. In the Six-Hump Camel Back function, where the optima are very close to each other, the parameter is responsible for the small steps necessary not to escape them. Parameter tuning methods as employed in 5.2 can help to exploit the newly added potential.

As for classification problems, since they are also generally n-dimensional, EGGC should be preferred again. Concerning accuracy improvement, a deeper study and preprocessing on the particular data set prior to applying the algorithm may be helpful. Parameter tuning alone seems unable to resolve this issue. Also, more work needs to be done on the hybridization of the evolutionary algorithm with the issues arising from classification. Last, a training-validation-testing procedure, as common in genetic programming, might lead to better results.

The major weakness of our algorithm lies in the need to find proper values for mating, merging and replacement radii. Radii self-adaptation is one of the main goals of future work. Another goal would be to try a non-generational selection for replacement strategy instead of the one we use and see whether is performs better or worse. A third goal would be to test if GC algorithms can be combined successfully with birth surplus-driven selection schemes as known from evolution strategies.

Acknowledgements. *The authors would like to thank the anonymous referees for the useful suggestions in improving this work.*

Bibliography

[1] D. Dumitrescu, "Genetic chromodynamics," *Studia Universitatis Babes-Bolyai Cluj-Napoca, Ser. Informatica*, vol. 45, no. 1, pp. 39–50, 2000.

[2] D. Dumitrescu and R. Gorunescu, "Evolutionary clustering using adaptive prototypes," *Studia Universitatis Babes-Bolyai Cluj-Napoca, Ser. Informatica*, vol. 49, no. 1, pp. 15–20, 2004.

[3] R. Gorunescu and P. H. Millard, "An evolutionary model of a multidisciplinary review panel for admission to long-term care," in *Proc. of ICCC 2004*, I. Dzitac, T. Maghiar, and C. Popescu, Eds., 2004, pp. 181–185.

[4] C. Stoean, R. Gorunescu, M. Preuss, and D. Dumitrescu, "An evolutionary learning spam filter system," in *Proceedings 6th International Symposium, SYNASC04 - Symbolic and Numeric Algorithms for Scientific Computing*, D. Petcu, D. Zaharie, V. Negru, and T. Jebelean, Eds. Timisoara, Romania: Mirton Publishing House, 26 - 30 September 2004, pp. 512–522, iSBN 973-661-441-7.

[5] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 207–234, 2002.

[6] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proceedings of 1996 IEEE International Conference on Evolutionary Computation (ICEC '96), Nagoya*, T. Fukuda, T. Furuhashi, and D. B. Fogel, Eds. Piscataway NJ: IEEE Press, 1996, pp. 798–803.

[7] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. 3rd Int'l Conf. on Genetic Algorithms ICGA 89*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 42–50.

[8] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, University of Illinois at Urbana Champaign, 1995.

[9] F. Streichert, G. Stein, H. Ulmer, and A. Zell, "A clustering based niching method for evolutionary algorithms," in *Genetic and Evolutionary Computation – GECCO-2003*, E. Cantú-Paz, Ed. Berlin: Springer-Verlag, 2003, pp. 644–645.

[10] O. M. Shir, "Niching in evolution strategies," in *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'05*, 2005, accepted.

[11] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[12] K.-Y. Lee and M. I. Roh, "An efficient genetic algorithm using gradient information for ship structural design optimization," *Journal of Ship Technology Research*, vol. 48, no. 4, pp. 161–170, 2001.

[13] D. Zaharie, "Extensions of differential evolution algorithms for multimodal optimization," in *Proc. of SYNASC'04*, D. Petcu, V. Negru, D. Zaharie, and T. Jebelean, Eds. Timisoara: Mirton, 2004, pp. 523–534.

[14] D. Beasley, R. Bull, and R. R. Martin, "Sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101–125, 1993.

[15] V. S. Gordon and L. D. Whitley, "Serial and parallel genetic algorithms as function optimizers," in *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 177–183.

[16] T. Bartz-Beielstein, K. E. Parsopoulos, and M. N. Vrahatis, "Design and analysis of optimization algorithms using computational statistics," *Applied Numerical Analysis & Computational Mathematics (ANACM)*, vol. 1, no. 2, pp. 413–433, 2004.

[17] T. Bartz-Beielstein, "New experimentalism applied to evolutionary computation," Ph.D. dissertation, University of Dortmund, 2005.

[18] L. Prechelt, "Proben 1 - a set of benchmarks and benchmarking rules for neural network training algorithms," University of Karlsruhe; Institute for Program Structures and Data Organization (IPD), Tech. Rep. 21/94, 1994.

[19] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," in *Proceedings of 12th Symposium on Computer Applications in Medical Care*, R. A. Greenes, Ed. IEEE Computer Society Press, 1988, pp. 261–265.

[20] W.-H. Au and K. C. C. Chan, "Classification with degree of membership: A fuzzy approach," in *Proc. of the 1st IEEE Int'l Conference on Data Mining*, San Jose, CA, 2001.

[21] R. Smithies, S. Salhi, and N. Queen, "Adaptive hybrid learning for neural networks," *Neural Computation*, vol. 16, no. 1, pp. 139–157, 2004.

[22] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, 1997.